Grado en Ingeniería Telemática

Trabajo de Fin de Grado

Desarrollo de una aplicación de guía turístico avanzado para dispositivos móviles Android

Autor: Iván Tacero Pasamontes

Tutor: Dr. David Griol Barres

Leganés, septiembre de 2017

Título: Desarrollo de una aplicación de guía turístico avanzado para dispositivos móviles Android

Autor: Iván Tacero Pasamontes

Tutor: Dr. David Griol Barres

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario : _____

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A mis compañeros y amigos de clase durante estos años, con los que he compartido momentos dentro y fuera de la Universidad, en especial a mi mejor amigo y compañero Gabriel, con el que he recorrido este largo camino que llega ya a su fin, a la espera de que nuevas puertas se abran para conseguir y superar nuevos metas y retos.

A mis padres, Ana y Cristóbal, porque sin ellos no hubiera sido posible llegar hasta aquí, y siempre me han apoyado para dar lo mejor de mí en los momentos más difíciles a lo largo de estos años. Sin olvidarme de mi novia, Beatriz, que ha sido un pilar fundamental en mi último curso.

Por último, a mi tutor David, que ha estado muy pendiente durante el desarrollo del proyecto. Sin olvidarme de mis compañeros del trabajo, con los que convivo diariamente y me ayudan a crecer tanto profesional como personalmente.

GRACIAS.

Resumen

Hoy en día los móviles se han vuelto indispensables en nuestro ritmo de vida actual, sobre todo cuando viajamos a lugares que no conocemos y surge la necesidad de saber la localización en la que nos encontramos o como llegar a un lugar determinado. Pero ¿qué pasa cuando viajamos a nuevos lugares y queremos visitar sus puntos más emblemáticos? El origen de esta necesidad me dio la idea de desarrollar una aplicación que nos proporcionará dicho servicio.

Una vez compartida la idea con el tutor y contar con su aprobación, inicie el desarrollo de esta. En primer lugar fue comprobar cómo estaba el mercado de aplicaciones que ofrecieran servicios similares a los que tenía pensados, tras comprobar el mercado, encontré varias aplicaciones de las cuales adopte la idea de cómo implementar la funcionalidad de las rutas turísticas. El resto de funcionalidades añadidas a la aplicación desarrollada en el presente proyecto, como la de buscar lugares de interés cercanos a un punto o de cómo llegar hasta ellos, se hizo con la idea principal de mejorar o completar los servicios de las aplicaciones existentes.

Tras explorar el mercado tocó definir el conjunto de funcionalidades requeridas, entre las que estaban implementar un guía turístico capaz de mostrar rutas en un mapa y reproducir mediante un audio la información de cada uno de los puntos de la ruta. Además de que una vez completada la ruta se dejaría al usuario introducir la opinión sobre esta y realizar el análisis de sentimientos de la misma, todo esto en cuanto a requerimientos funcionales. En cuanto a requerimientos de diseño, la idea estaba en crear una aplicación con una interfaz gráfica seria, sencilla, agradable e intuitiva, con solo los botones que fuesen necesarios y dejando mayor espacia posible al mapa de la cuarta actividad.

Con los objetivos que se pretendían alcanzar con el desarrollo de la aplicación bien definidos, y teniendo en cuenta mi inexperiencia en el desarrollo de aplicaciones Android, los inicios no fueron fáciles y los problemas no tardaron en aparecer. El principal problema fue en encontrar una versión del sistema operativo Android con la que poder compilar un proyecto con Google Maps en Android Studio. Finalmente, tras varias pruebas, se creó el proyecto con la versión 5.0 y el emulador para ejecutar con la 5.1

Seguidamente comencé con el estudio de las APIs de Google, en combinación con pruebas para aprender a usarlas, sobre todo en como deserializar las respuestas que estás devolvían. Tras esto, se empezó con la implementación de la aplicación, en concreto de la funcionalidad principal, el mostrado de rutas, qué incluía la llamada a Google Directions API. Una vez conseguido, se creó la primera de las tres tablas de base de datos, routes.sqlite, tabla para trabajar con la información de la ruta. Las otras dos tablas son, PointTracked.sqlite, que guarda información del usuario y la última, NamePointText.sqlite, esta guarda la descripción de cada uno de los puntos.

Para completar la actividad principal solo quedaba la reproducción y el control de archivos de audios, con esto se completaba casi toda la funcionalidad básica. Solo faltaba implementar el uso del análisis de sentimientos, para ello se necesitó guardar cada uno de los puntos que el usuario recorría, y cuando este completara todos accedería a la función de introducir su opinión y el posterior análisis, con el uso de la API de MeaningCloud

Cuando los requisitos funcionales se completaron, se añadió a la aplicación una nueva funcionalidad, la búsqueda de lugares cercanos de interés y el cálculo de ruta hacia ellos, para esto se usó Google Places. A parte, se añadieron nuevas actividades que mejoraban la experiencia de usuario, como la posibilidad de buscar rutas en función de unas preferencias establecidas por el usuario y una actividad para mostrar un conjunto de información de la ruta preelegida .

Una vez que conseguí ensamblar todas las funcionalidades anteriores de forma robusta, viendo que contaba con tiempo y ganas, creé en la nube de Microsoft Azure una plataforma de procesos de negocios automatizados. Esta es capaz de procesar los tweets bajo un # y publicar en un canal de Slack según el contenido de estos sea malo, bueno o muy bueno.

Finalmente añadir que, los objetivo marcados en un principio se han cumplido, consiguiendo adquirir experiencia y manejo en el desarrollo de aplicaciones Android. Además de conocer plataformas Cloud como Azure, que abre un abanico de posibilidades que aumentan el número de funcionalidades que la aplicación puede ofrecer.

Palabras clave: Aplicaciones móviles, Sistema Operativo Android, Guiado turístico, APIs de Google, Análisis de sentimientos, MeaningCloud, Microsoft Azure.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
INTRODUCCIÓN

Este documento describe el proyecto realizado como Trabajo Fin de Grado, el cual consiste en el desarrollo una aplicación de guía turístico para el sistema operativo Android, que ofrece al usuario la posibilidad de buscar rutas turísticas según sus preferencias, reproducir audios con la información de cada punto de la ruta, localizar puntos de interés cercanos a los puntos de la ruta, calcular la ruta hacia alguno de estos puntos y opinar de forma oral sobre el contenido de cada ruta.

A continuación, se introducen las tecnologías utilizadas y se destacan los aspectos que han motivado la realización del programa como una aplicación Android.

1.1 Dispositivos móviles

La evolución de smartphones y tablets ha cambiado los hábitos de consulta en la red. Gracias a las redes inalámbricas los usuarios pueden consultar contenidos online desde cualquier lugar, esto supone tener acceso a la información y servicios en todo momento y permite al usuario un uso totalmente espontáneo del dispositivo eliminando la necesidad de planificación que implicaba el tener que consultar los datos de interés desde redes domésticas o de trabajo.

Los dispositivos móviles han cambiado la manera en la que los usuarios acceden a la información, y esto ha sido principalmente debido al acceso a las redes móviles y al cambio de concepto que introdujeron las aplicaciones.

1.1.1 Redes inalámbricas

La independencia de los cables ha aumentado enormemente la movilidad de los usuarios y con ella también han aparecido multitud de nuevos servicios y usos de la red de datos. Como refleja un estudio de Cisco sobre el impacto del tráfico de datos móviles [CISCO], el tráfico móvil a nivel mundial crece muy rápido, y las previsiones indican que continuará la misma tendencia. Este estudio desvela como el incremento exponencial de usuarios móviles, smartphones y conexiones del Internet of Things (IoT), junto a las mejoras en velocidad de red y el mayor consumo de vídeo móvil, multiplicarán por siete el tráfico de datos móviles en los próximos cinco años.

Previsiones de Cisco:

- **El tráfico global de datos móviles representará el 20 por ciento del tráfico IP total** (desde el 8 por ciento que suponía en 2016).

- **Habrá 1,5 dispositivos móviles por persona.** Casi12.000 millones de dispositivos móviles conectados -incluyendo módulos máquina-a-máquina (M2M, Machine-to-Machine) para una población mundial estimada de 7.800 millones de habitantes.

- **La velocidad media de las redes móviles se multiplicará por tres** desde los 6,8 Mbps en 2016 hasta los 20,4 Mbps en 2021.

- **Las conexiones M2M supondrán el 29 por ciento (3.300 millones) del total de conexiones móviles**, desde el 5% que representaban en 2016 (780 millones). M2M será el tipo de conexión móvil de más rápido crecimiento como resultado del incremento de aplicaciones del Internet of Things (IoT).

- **Las redes 4G soportarán el 58 por ciento del total de conexiones móviles en 2021** (26 por ciento en 2016) y generarán el 79 por ciento de todo el tráfico de datos móviles.

Otras previsiones adicionales según Cisco, es que el tráfico global de datos móviles alcanzara los 49 Exabytes mensuales, lo que supone un ratio de incremento interanual de 47% entre 2016 y 2021. Por otro lado el video móvil se multiplicará por 8,7 entre los mismo años. Para finalizar añadir el aumento de la VR y AR. La primera de ella incrementará su tráfico 11 veces entre 2016 y 2021 y la segunda de ellas por 7.

La Figura 1 refleja los datos comentados en el párrafo anterior: muestra la estimación de crecimiento exponencial del tráfico móvil en el periodo de tiempo estudiado.

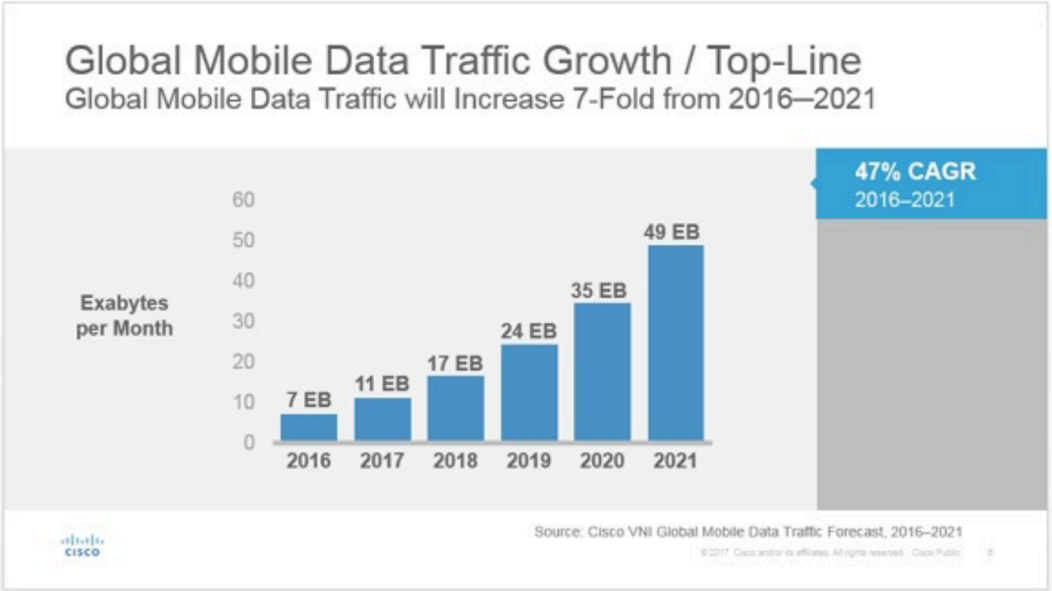


Figura 1: Previsión del consumo de datos móviles [CISCO]

1.1.2 Aplicaciones móviles

Una mayor parte del contenido consumido ya no se hace a través de webs sino de aplicaciones. Las aplicaciones móviles como las conocemos hoy en día nacieron con el lanzamiento del iPhone de Apple, y crecieron enormemente en un corto periodo de tiempo en las plataformas iOS y Android sobre todo, gracias al impulso de las tiendas de aplicaciones y de las facilidades de que disponen los desarrolladores.

Las aplicaciones proporcionan muchas ventajas respecto a las consultas web tradicionales. Algunos ejemplos son la posibilidad de utilizar funciones nativas del teléfono, acceder a la información deseada inmediatamente sin pasar por filtros o búsquedas, disponer de contenido offline y la posibilidad de personalizar el aspecto y el funcionamiento. Cada vez están más especializadas y desempeñan funciones concretas de manera eficaz y creativa, son atractivas y abarcan cualquier ámbito, desde el mero entretenimiento hasta ser utilizadas como herramientas de trabajo. Hay una buena oferta de aplicaciones para cada posible uso del dispositivo, y han cambiado el modo en el que el usuario interacciona con ellos.

La Figura 2 muestra unas estadísticas [VPNMENTOR] del tiempo empleado por los usuarios en aplicaciones móviles en comparación con el que emplean en navegadores web. Los usuarios de dispositivos móviles pasan el doble

de tiempo en aplicaciones que en páginas web para móviles. En los años futuros, se espera que la brecha se amplíe aún más.

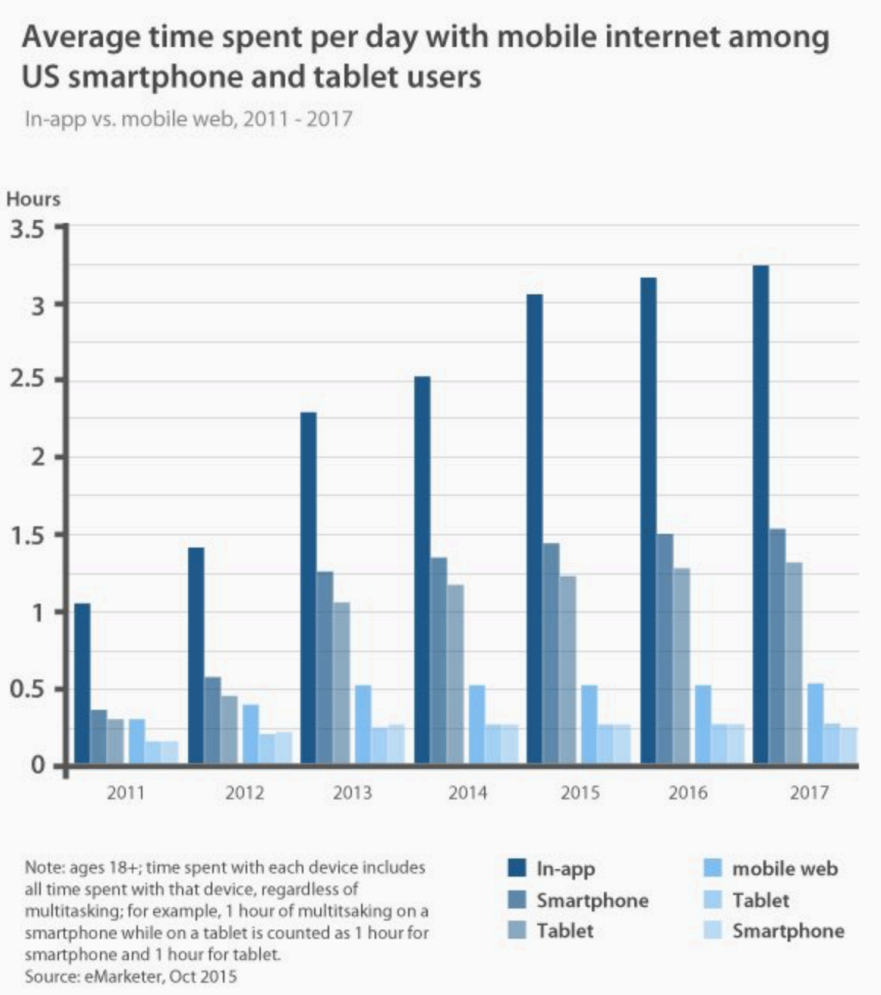


Figura 2: Tiempo empleado en aplicaciones y navegadores web [VPNMENTOR]

Android

Android es una solución software completa para dispositivos móviles que engloba el sistema operativo (basado en Linux), un entorno de ejecución basado en Java, librerías de bajo y medio nivel y un conjunto inicial de aplicaciones para el usuario final. Y es un software open-source, lo cual implica que nadie en la industria puede restringir o controlar las innovaciones de cualquier desarrollador. Se le permite explorar la plataforma y modificarla en función de sus necesidades y, por supuesto, contribuir al proyecto open-source mostrando esos cambios a la comunidad en el ecosistema Android [SOURCEANDROID].

En la Figura 3 [STATISTA], podemos ver la evolución de la cuota de mercado de los diferentes sistemas operativos desde el 2009 hasta el 2016. Donde destaca el crecimiento de dispositivos Android entre los años 2009 y 2013, llegando a abarcar el 75% del mercado. En este último año su crecimiento se ha normalizado y se mantiene más o menos constante. Por otro lado la evolución de su competidor más directo iOS tiene una curva de cuota de mercado constante entre el 15% y 25% con pequeñas picos de bajada y subida.

En cuanto al desarrollo de aplicaciones, cada día es más fácil gracias a la gran comunidad de desarrolladores, código accesible en la web, ibrerías que cubren tanto las necesidades básicas como las más avanzadas y la compatibilidad entre versiones gracias al Support Package, aunque esta no es total. Por ello a la hora de crear una aplicación se ha de elegir el nivel de API, para esta elección consultar interesante consultar desde qué nivel de API se han realizado los accesos a Google Play Store (la tienda de aplicaciones) [DEVELOPERANDROID]. Según la Figura 4, parece razonable usar como versión mínima soportada por la aplicación la 5.0.

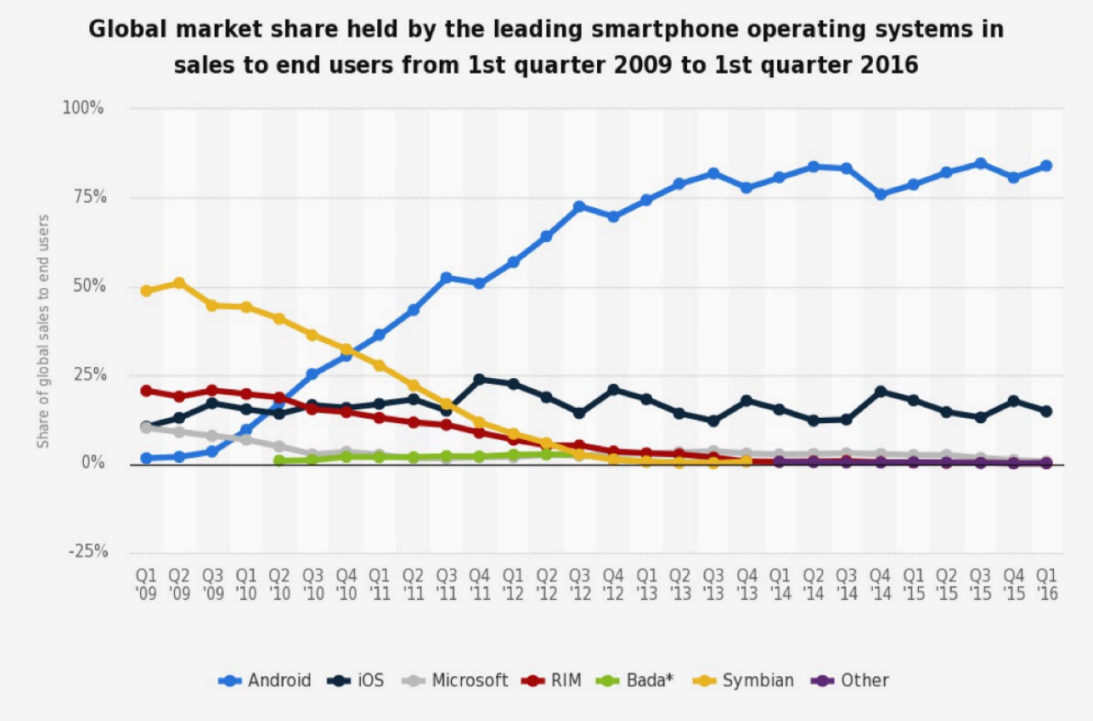


Figura 3: Previsión del consumo de datos móviles [STATISTA]

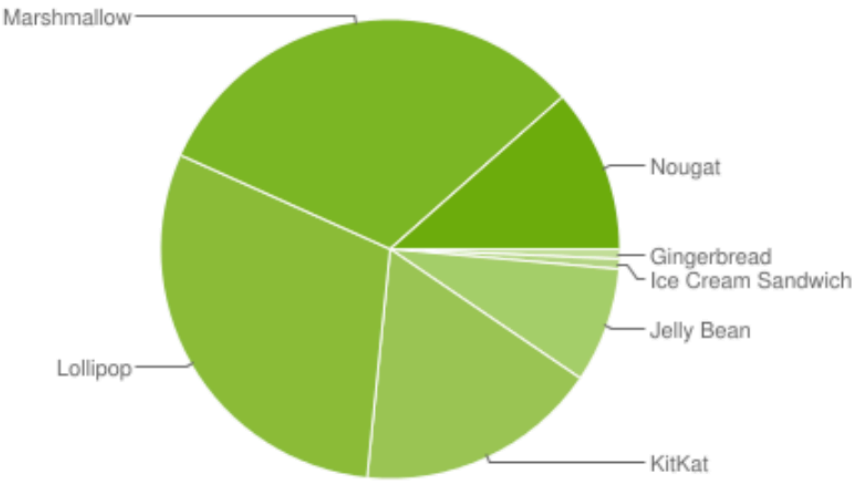


Figura 4: Acceso a Google Play por versiones del S.O [DEVELOPERANDROID]

1.2 Objetivos

Este apartado contiene dos secciones fundamentales, por un lado, describe los objetivos del Trabajo Fin de Grado elegido, y por otro, los objetivos específicos de la aplicación desarrollada.

1.2.1 Objetivos del Trabajo de Fin de Grado

En la definición del Trabajo de Fin de Grado elegido se requería desarrollar una aplicación que usara realidad aumentada en combinación con un servicio de análisis de sentimientos. Debido a que las elecciones sobre la funcionalidad y el diseño de la aplicación eran totalmente libres, se decidió cambiar el requisito de realidad aumentada por la introducción de *APIs* de Google Maps y reproducción de archivos multimedia como audios y manejo de base de datos *Sqlite*. A continuación, se exponen los objetivos del proyecto.

- **Dominar el entorno de desarrollo**: es importancia estar familiarizado con el *SDK* y con la programación en Android. A parte de conocer la distribución de archivos y carpetas del proyecto, qué principalmente se

dividen en archivos *JAVA* para la lógica y *XML* para las vistas y ficheros de configuración.

- **Manejo y conocimiento de las *APIs***: profundizar en el estudio de los servicios que Google ofrece para el trabajo con mapas. A parte de las APIs de Google, también se ha usado *MeaningCloud*, servicio necesario para realizar el análisis de sentimiento de las opiniones de los usuarios.

- **Uso de base de datos *Sqlite***: entre los objetivos también se propuso utilizar bases de datos *Sqlite* para el guardado de información, como la propia de la aplicación como la correspondiente al usuario.

- **Azure**: introducción en la nube de Azure para implementar flujos de trabajos automatizados. En mi caso destinados a procesos de negocio.

1.2.2 Objetivos de la aplicación

Teniendo en cuenta la inexperiencia del alumno en el desarrollo de aplicaciones Android, seguidamente se van a detallar los objetivos para esta aplicación. Ésta consiste en un guía turístico capaz de representar rutas turísticas en mapas en combinación con la reproducción de audios sobre la información de los puntos que componen la ruta.

- **Desarrollar una aplicación útil per sencilla y robusta**: se pretende implementar una aplicación cuya funcionalidad principal, representación de ruta en un mapa y reproducción de audios con la información de los puntos de la ruta, funcione de manera robusta y sin fallos.

- **Apariencia agradable**: diseñar una aplicación visualmente seria, sencilla y agradable, con los botones necesarios.

- **Facilidad de uso e intuitiva**: implementar una aplicación en la que cada actividad aísle una funcionalidad concreta, además de añadir a cada una de estas subelementos para mejorar la experiencia de usuario.

- **Adicción de funcionalidades extras**: se han añadido funcionalidades extras que potencian los servicios ofrecidos por la aplicación, cómo la posibilidad de búsqueda de puntos de interés cercanos a un punto de la

ruta seleccionado, y cálculo de la ruta hacia el mismo.

- **Procesos de negocio**: aparte del desarrollo de la aplicación, he creado una plataforma en la nube de *Azure* para automatizar procesos de negocio, con la idea de hacer un seguimiento automatizado sobre lo que se publica en Twitter en referencia al funcionamiento de la aplicación.

1.3 Planificación

Al planificar la realización de proyecto se han definido tres fases principales:

- **Fase de planificación**: en esta primera etapa se ha realizado un estudio de las posibilidades que el sistema operativo Android ofrece para cumplir los objetivos marcados. Además de hacerse una comparación entre las diferentes versiones del sistema operativo para elegir una compatible con las *APIs* de *Google* utilizadas.

- **Fase de desarrollo**: en esta etapa se ha implementado y evaluado la aplicación realizando numerosas pruebas y modificaciones.

- **Fase de documentación**: realización de la memoria del Trabajo Fin de Grado. La totalidad de esta fase ha sido tras finalizar la implementación de la aplicación.

- **Generación del contenido**: crear el contenido propio de la aplicación. Por una parte, los audios en formatos *Ogg* con la información de cada punto de la ruta. Y por otro lado, la información equivalente en formato texto.

- **LogicApps**: esta última fase se dedicó a la creación de una plataforma en *Azure* para procesos de negocios automatizados

1.3.1 Fase de planificación

- **Definición de los objetivos**: a medida que el proyecto avanzaba, los objetivos y expectativas han ido sufriendo modificaciones ya que se descubrían nuevas posibilidades y se descartaban algunas de las

anteriormente planteadas por su dificultad o utilidad. Esta fase por tanto abarca las reuniones con el tutor para definir los objetivos finales.

- **Elección de la versión y configuración del emulador**: antes del desarrollo ha sido necesario realizar diversos tipos de prueba para la elección de la versión del sistema operativo Android. Para evitar problemas a la hora de la compilación, y tras diversas pruebas, el proyecto en *AndroidStudio* se creó sobre la versión 5.0, mientras que en la configuración del emulador se instaló la 5.1.

- **Documentación sobre las APIs a usar**: proceso de investigación inicial sobre las *APIs* de *Google* que se habían pensado usar. *APIs* muy fáciles de llamar, pero con un alto grado de complejidad a la hora de mapear la respuesta de esta, en especial la API de *Google Direcctions*. A parte, para la implementación del servicio de análisis de sentimientos se ha usado la *API* de *MeaningCloud*.

- **Documentación sobre las LogicApp:** análisis e investigación de flujos de trabajo en la nube de *Azure*, para implementar procesos automatizados.

1.3.2 Fase de desarrollo

Posteriormente se detalla de una forma resumida las etapas por las que se ha pasado durante la implementación del código.

- **Creación del proyecto y configuración emulador**: el primer paso fue crear de un proyecto de prueba sobre *AndroidStudio,* usando una plantilla que incorpora ya por defecto una instancia de un objeto *GoogleMap.* Después se realizó la configuración del emulador basado en un Nexus 5 con la versión 5.1 de Android para poder correr la aplicación.

- **Mostrar la ruta en el mapa**: una vez que el mapa está operativo, se hace uso de la API *Google Direcctions* para obtener las coordenadas de la ruta y añadirlas al mapa.

- **Primera tabla de base de datos Sqlite**: la estructura se puede ver en la Figura 7 del capítulo 3. En ésta se guardan datos referentes a las rutas que contiene la aplicación.

- **Reproducción de audios y cuadro de controles**: A través de la clase *MediaPlayer* se gestiona las acciones (Play, Stop) sobre los archivos de audio con la información de cada uno de los puntos. A parte y como funcionalidad extra, se diseñó una *progressBar* implementada bajo una *AsyncTask,.*
- **Almacenamiento dinámico de puntos recorridos por el usuario**: se crea otra tabla de datos, cuya estructura puede verse en la Figura 17, del Capítulo 3. En esta se va guardando de forma dinámica cada uno de los puntos de la ruta que el usuario va seleccionando.

- **MeaningCloud análisis de sentimientos**: cuando el usuario ha completado la ruta este podrá acceder a la última Actividad, y de forma oral podrá decir su opinión, qué se enviará en formato texto a la API, y así obtener la nota del análisis.

- **Actividad filtros para búsqueda de rutas**: para mejorar la experiencia de usuario se añadió una actividad que permitiera realizar búsqueda de rutas en función de unas preferencias que el usuario elija. Esta Actividad se corresponde con la Figura 9.

- **Actividad resumen de la ruta preelegida**: otra actividad que se añade para potenciar la funcionalidad de la aplicación, es la que corresponde a la Figura 11. En ella el usuario puede ver información de distinto tipo de la ruta elegida, como fotos, un resumen o las opiniones de los usuarios.

  Tabla para almacenar las descripciones de los puntos de la rutas: al pulsar el botón *Info* que se ve en la Figura 16, se despliega una ventana en la que aparecerá el nombre del punto actual seleccionado seguido de la información. Esta información esta guardada en la tabla *NamePointText.sqlite.*

- **Búsqueda de lugares de interés y ruta hacia ellos** : otra funcionalidad extra que muestra en el mapa la situación de los puntos con un icono representativo de este, y permite al usuario buscar en Internet información sobre el lugar o calcular la ruta hacia él.

- **Generación del contenido propio de la aplicación**: una vez que se había alcanzada de forma estable los diferentes flujos de funcionalidades de la aplicación, se terminó por añadir el contenido necesario por la aplicación. Entre este contenido diferenciamos:

1. Tabla *routes.sqlite*, Figura 7: a parte de la entrada de ejemplo, se han añadido nuevas entradas para poder visualizar otras rutas a parte de la del ejemplo.

2. Tabla *NamePointText.sqlite,* Figura 16: insertar la descripción de cada punto para cada una de las rutas que contiene la aplicación.

3. Creación y almacenado de audios: creación de audios en formato *mp3* uno por cada punto de cada ruta y con la información contenida en la tabla anterior. Una vez creados todos los audios se convertirán a formato *Ogg,* para ello se usará el programa Switch.

- **Logic Apps:** completadas todas las etapas del desarrollo de la aplicación, creé en *Azure* una plataforma que provee de un servicio de procesos de negocio de forma automatizada. Esta plataforma se basa en una *Logic App* que se ejecuta cada un tiempo x configurado. Tras ejecutarse procesa los tweets para finalmente publicarlos junto con su firma en el canal de *Slack* correspondiente (malo, bueno o muy bueno) según el análisis de sentimientos del contenido del tweet.

1.3.3 Fase de documentación

Durante las fases de planificación y desarrollo del programa se ha ido registrando el proceso seguido y la información recopilada sobre los recursos empleados. Al finalizar las dos primeras etapas se ha redactado en esta memoria la información acumulada con referencias a las fuentes de información y bibliografía utilizadas.

1.4 Medios empleados

En este apartado se detallan las utilidades de software, dispositivos, hardware y lenguajes de programación empleados en el proceso de desarrollo de la aplicación, y también los manuales consultados para la ayuda y el aprendizaje.

1.4.1 Lenguajes de programación

En esta aplicación Android se ha utilizado *XML* para diseñar la parte gráfica de todas las ventanas y actividades, mientras que con Java se ha programado todo la lógica de la aplicación.

Por último para el desarrollo de la *Logic App* se ha usado el portal de *Azure*, y para completar el flujo de trabajo de la aplicación lógica se ha implementado una *Azure Function* en C#, también en el propio portal.

1.4.2 Software

Se han utilizado los siguientes elementos:

- **Android Studio** con la versión 2.2.2, es el entorno de desarrollo en el que se ha implementado todo el código de la aplicación. Nos ofrece una gran cantidad de posibilidades de configuración, entre algunas de ellas está la posibilidad de instalarse la versión de Android que se prefiera con la ayuda del *Android SDK* y configurarse un emulador con el *ADV Manager* para poder ejecutar y probar la aplicación. También nos da la posibilidad a través de *Android Device Monitor* de poder acceder y modificar cualquier carpeta interna del dispositivo en ejecución. Esto ha sido útil a la hora de crear las bases de datos sqlite.

- **Dispositivo móvil** (emulador) Google LG *Nexus 5* para ejecutar y depurar la aplicación.

- **SQLite Manager** versión 0.8.3.1, es un complemento de *Firefox*. Se ha usado para el manejo de base de datos sqlite, principalmente a la hora de poder ver el contenido almacenado y realizar modificaciones sobre este.

- **Switch** versión 5.15, este programa se ha usado para poder convertir los audios de formato *Mp3* a *Ogg*. Este formato es más adecuado para almacenar archivos multimedia, audios en este caso, para que ocupen menos espacio en la memoria del dispositivo.

- **Portal de Azure** se ha usado para implementar todo el flujo de trabajo necesario de la Logic App.

- **Android 5.1** en el emulador para ejecutar y depurar la aplicación.

- **Microsoft Office 365** con *Microsoft Word* actualizado a la versión 2016 para redactar toda la fase de documentación.

- **Gantt Proyect** para el diagrama de Gantt con la planificación temporal.

1.4.3 Hardware

- **Ordenador portátil** *MacBookPro* con *macOS Sierra* versión 10.12.3 para el desarrollo del TFG.

1.4.4 Manuales

La web *'Developers'* [DEVELOPERANDROID] de Android contiene información sobre todas las funciones y librerías disponibles, guías sobre las *APIs* y ejemplos para aprender y entrenar. Además de esta web de referencia, en Internet hay numerosos tutoriales con ejemplos para casi para todo. Al final de la memoria en el apartado *'Referencias'*, se han añadido las más relevantes.

También ha sido imprescindible la web 'Developers' [GOOGLEDEVELOPERS] de *Google*, ésta contiene documentación entre otras cosas acerca de todas las *APIs* que *Google* ofrece. A parte, es necesario acceder a esta con tu cuenta *Google* para poder conseguir las credenciales para usar sus *APIs* y registrar el nombre de tu proyecto desde donde van a ser usadas. Para poder usar estas *APIs* se han de habilitar desde el panel de control dentro de esta web.

Otro de los servicios externos utilizados en este proyecto ha sido la *API* de *MeaningCloud* [MEANINGCLOUD], para en análisis de sentimiento de las opiniones de los usuarios.

Para la implementación de la *Logic App* en el portal de *Azure* me ha resultado imprescindible los '*Docs'* [MICROSOFT] de Microsoft, web donde tienen unificada toda la documentación técnica de todos sus productos.

Por último, la web *Stack Overflow* [STACKOVERFLOW], qué permite hacer preguntas a su gran comunidad de usuarios para que éstos publiquen sus soluciones. Prácticamente cualquier duda que pueda surgir ha sido preguntada allí, y aunque no siempre se pueda implementar la solución, es una gran fuente de inspiración.

1.5 Planificación temporal

El proyecto ha sido desarrollado desde diciembre de 2016 hasta el mes de Julio de 2017. Las primeras reuniones tuvieron lugar en noviembre, donde se intercambiaron ideas de conceptos de posibles aplicaciones en función del requisito de implementarla bajo el uso de *cognitive-services*.

Tras un par de reuniones se cerró la idea de implementar un guía turístico, basado en el uso de *APIs* de *Gooble Maps* y *cognitive-services.* Una vez se tenía la idea cerrada, se ha ido desarrollando la aplicación. Cuando los requisitos imprescindibles se alcanzaron, se fue pensando ideas que aumentarán el rango de funcionalidades que la aplicación ofrecía, para que el usuario tuviera una mejor experiencia. Por último, se creó una plataforma que provee de un servicio de procesos de negocio de forma automatizada.

En la siguiente página se muestra un diagrama de Gantt con la planificación temporal del proyecto.

1.6 Estructura de la memoria

Seguidamente se incluye un breve resumen de cada capítulo, con el objetivo de mostrar la manera en la que está estructurada:

Capítulo 1: Introducción y objetivos. Este capítulo inicial describe los objetivos del Trabajo Fin de Grado. También se explican las fases del desarrollo, los medios empleados y la planificación temporal, y se hace una breve descripción de la estructura de la memoria.

Capítulo 2: Estado del arte En este capítulo se hace un estudio del marco actual en cuanto a las diferentes características que presenta la aplicación. Se analizan el conjunto de *APIs* que *Google* ofrece para trabajar con mapas, junto a como el uso del Big Data ha promovido el desarrollo de servicios de inteligencia avanzada. En concreto para mi proyecto se ha hecho uso de los servicios cognitivos para el análisis de sentimiento de opiniones de usuarios.

Por otra parte, en este capítulo también analiza en marco actual en el que se encuentran las *Logic Apps* actualmente, para la creación de flujos de trabajos automatizados en la nube.

Tabla 1: Diagrama de Gantt con la planificación del proyecto

Capítulo 3: Descripción general del programa desarrollado. Se proporciona una visión global de la aplicación y se detallan las posibilidades que ofrece.

Capítulo 4: Evaluación de la aplicación. En este capítulo se muestra cómo ha respondido la aplicación a las pruebas realizadas durante la fase de desarrollo.

Capítulo 5: Conclusiones y trabajo futuro. Se exponen las ideas y conclusiones obtenidas tras la realización del proyecto y se analizan las líneas de investigación que a partir de este proyecto se podrían generar.

Capítulo 6: Presupuesto e impacto social y económico. El trabajo incluye un análisis detallado del impacto socio- económico y además incluye un presupuesto bien definido en todos sus conceptos.

Capítulo 7: Marco regulador. Este apartado incluye los estándares técnicos en los que queda enmarcado el proyecto.

Glosario. Este apartado recopila los principales acrónimos y términos técnicos utilizados en la memoria con el fin de facilitar la comprensión del lector.

Referencias. Este apartado muestra los documentos que se han consultado tanto durante la realización de la aplicación como de la memoria.

Capítulo 2

ESTADO DEL ARTE

Es importante que antes de desarrollar un proyecto se tenga una visión global de cómo éste quedaría enmarcado en el estado del arte y qué aportaría respecto a los demás.

En este caso el proyecto consiste en una aplicación para la plataforma móvil Android que emula el funcionamiento de guía turístico. Para ello, ha sido necesario trabajar con instancias *GoogleMap* a la hora de obtener mapas, para calcular las rutas se han realizado llamadas a la *API Google Direcctions,* y para la búsqueda de lugares cercanos se ha usado *Google Places*. Por otro lado, para la implementación de los servicios de análisis de sentimientos se ha trabajado con *MeaningCloud*.

En este capítulo se van a analizar los diferentes elementos necesarios para desarrollar una aplicación capaz de mostrar rutas y lugares cercanos en mapas. A parte de comprobar el punto en el que se encuentran las aplicaciones que se basan o incorporan en su funcionamiento servicios Machine-Learning. En última estancia, también se analizará la forma de crear plataformas en la nube para procesos automatizados junto con sus ventajas y posibles aplicaciones.

2.1 APIs para la programación y trabajo con Google Maps en Android

2.1.1 Introducción

Las aplicaciones que integran *Google Maps* ofrecen una serie de servicios al usuarios basados en la utilización de mapas. Principalmente, las aplicaciones incorporan mapas para ofrecer servicios que se basan en la ubicación del usuario.

A continuación, se van a detallar las librerías que *Google* ofrece y se han usado en el desarrollo de la aplicación para implementar las llamadas a las *APIs* y poder así trabajar con mapas.

2.1.2 Google Direcctions API

Toda la documentación sobre esta interfaz de programación de aplicaciones ha sido tomada de la web de desarrolladores de *Google* [GOOGLEAPIDIRECTION].

Introducción

El API de rutas de Google es un servicio que utiliza una solicitud *HTTP* para calcular rutas para llegar de una ubicación a otra. Se puede buscar qué ruta se seguiría en función del método de transporte utilizado: en transporte público, en coche, a pie o en bicicleta. En las rutas pueden especificar los orígenes, los destinos y los hitos de dos formas, o bien como cadenas de texto (por ejemplo, "Chicago, IL" o "Darwin, NT, Australia") o como coordenadas de latitud/longitud.

Límites de uso

Para usuarios normales, Google ofrece un máximo de 2.500 solicitudes de rutas al día, las cuales pueden incluir 23 hitos intermedios como máximo. Los clientes de *Google Maps* for Bussines pueden recibir hasta 100.000 solicitudes de rutas diarias, con 23 hitos en cada una.

Ha de tenerse en cuenta que las URL del API de rutas de Google no pueden superar los 2.048 caracteres (antes de la codificación de URL). Dado que algunas URL del servicio de indicaciones pueden incluir varias ubicaciones a lo largo de una ruta, se recomienda que se tenga en cuenta este límite a la hora de crear las URL.

Estas limitaciones no afectan al desarrollo de la aplicación y difícilmente afectarán al futuro de la misma dado que tendría que ser muy popular para superar las 2.500 solicitudes diarias.

Autentificación

El *API* de Google Direcctions utiliza una clave *API* key que ha de ser añadida en cada solicitud que se realice. A parte dicha API debe estar habilitada en el panel del administrador de *APIs* de google.

Solicitudes de ruta

Una solicitud del API de rutas tiene el siguiente formato:

```
https://maps.googleapis.com/maps/api/directions/outputFormat?parameter
```

En esta solicitud, output debe ser uno de los valores que se indican debajo.

- json(recomendado) indica el formato de salida en Notación de objetos JavaScript (JavaScript Object Notation, *JSON*).

- xml indica el formato de salida como un archivo *XML*.

Parámetros de solicitud

Algunos parámetros son obligatorios y otros opcionales, como en las *URL* estándar, todos los parámetros se separan con el carácter &. Posteriormente se indican los parámetros más importantes y sus posibles valores.

Parámetros obligatorios

- origin: define la dirección o el valor de latitud/longitud textual de la ubicación desde la que se quiere calcular las rutas.

- destination: define la dirección o el valor de latitud/longitud textual de la ubicación desde la que se quiera calcular las rutas.

- Key: identifica tu aplicación a los fines de la administración de la cuota.

Parámetros opcionales

- mode (el valor predeterminado es walking): especifica el medio de transporte que se utilizará al calcular las indicaciones.

- waypoints: especifica un conjunto de hitos. Para modificar una ruta los hitos establecen las ubicaciones específicas por las que debe pasar. Ha sido utilizado en este proyecto, se han puesto las coordenadas de cada uno de los puntos turísticos que componen la ruta.

- alternatives: si se establece en true, indica que el servicio de rutas puede devolver más de una ruta alternativa. Tener en cuenta que la obtención de rutas alternativas puede incrementar el tiempo de respuesta del servidor. Este campo no se ha usado.

- avoid: Permite evitar autopistas y/o peajes.

- units: especifica el sistema de unidades que se utilizará para mostrar los resultados. Utiliza por defecto el sistema métrico oficial en el país de origen.

- departure_time y arrival_time especifican la hora de salida y llegada para las indicaciones en transporte público respectivamente. No han sido utilizadas.

Seguidamente se muestra un ejemplo de solicitud que permite obtener indicaciones de Toronto (Ontario) a Montreal (Quebec), en bicicleta y evitando las principales autopistas:

```
https://maps.googleapis.com/maps/api/directions/json?origin=Toronto&destination=Montreal&key=YOUR_API_KEY
```

Respuestas de rutas

Las respuestas de ruta se devuelven en el formato que indica la marca de output en la ruta de la solicitud de la URL, en nuestro caso JSON. Debajo se muestra un ejemplo para calcular la ruta desde Chicago, IL a Los Ángeles, CA que atraviese dos hitos en Joplin, MO y en Oklahoma City, OK:

```
https://maps.googleapis.com/maps/api/directions/json?origin=Chicago,IL&destination=Los+Angeles,CA&waypoints=Joplin,MO|Oklahoma+City,OK&key=YOUR_API_KEY
```

El resultado JSON se muestra a continuación dado que los resultados de rutas pueden demasiado grandes y sólo se quiere mostrar el formato, algunos elementos se han omitido.


JSON
geocoded_waypoints
routes
0
bounds
northeast
southwest
copyrights : "Datos de mapas ©2017 Google, Inst. Geogr. Nacional"
legs
0
distance
duration
end_address : "Plaza Emperador Carlos V, 8, 28012 Madrid, España"
end_location
lat : 40.4082257
lng : -3.6927818
start_address : "Ronda de Atocha, 2, 28012 Madrid, España"
start_location
lat : 40.4074393
lng : -3.6941989
steps
0
distance
duration
end_location
html_instructions : "Dirígete al **este** por **Ronda de Atocha** hacia **Plaza Emperador Carlos V**"
polyline
start_location
travel_mode : "WALKING"
1
traffic_speed_entry
via_waypoint
1
2
3
4
5
overview_polyline
summary : "Ronda de Atocha y Plaza Emperador Carlos V"
warnings
waypoint_order
status : "OK"

Por lo general, sólo se devuelve una entrada en el conjunto de "routes" para búsquedas de indicaciones. Sin embargo, es posible que el servicio de rutas devuelva varias rutas si se transmite alternatives=true.

Códigos de estado

El campo "status" del objeto de respuesta de rutas contiene el estado de la solicitud y puede incluir información sobre depuración para ayudar a descubrir el motivo por el que no funciona el servicio de rutas. El campo "status" puede contener los siguientes valores:

OK, NOT_FOUND, ZERO_RESULTS, MAX_WAYPOINTS_EXCEEDED, INVALID_REQUEST, OVER_QUERY_LIMIT, REQUEST_DENIED y UNKNOWN_ERROR.

Rutas

Cuando el *API* de rutas devuelve resultados los ubica en un conjunto de routes (*JSON*). Aunque el servicio no devuelva ningún resultado (por ejemplo, si no existe la dirección), el *API* devolverá un conjunto de routes vacío (las respuestas *XML* están formadas por cero o varios elementos <route>).

Cada elemento del conjunto de routes contiene un resultado único del origen y del destino especificados. Esta ruta puede constar de uno o varios legs, en función de los hitos que se hayan especificado. Además, la ruta también incluye información de derechos de autor y advertencias que se deben mostrar al usuario, junto con la información de la ruta.

Cada ruta del campo routes puede contener los siguientes campos:

- summary contiene una breve descripción textual sobre la ruta, que permite identificarla y distinguirla de otras alternativas.

- legs[] contiene cada tramo de la ruta definido por hitos. Se presentará un tramo diferente por cada hito o destino especificado (Una ruta sin hitos contendrá exactamente un tramo dentro del conjunto de legs). Cada tramo consta de una serie de pasos (steps). Se detalla más adelante.

- waypoint_order indica el orden de los hitos de la ruta calculada.

- overview_polylinecontiene un objeto que consta de un conjunto de puntos (points) codificados que representan una ruta aproximada (suavizada) de las indicaciones resultantes.

- bounds contiene el cuadro delimitador de la ventana gráfica de esta ruta.

- copyrights contiene el texto de los derechos de autor que se mostrará con la ruta.

- warnings[] contiene un conjunto de advertencias que se visualizará cuando se muestren las rutas.

Tramos (legs)

Cada elemento del conjunto de legs especifica un tramo único del trayecto desde el origen al destino de la ruta calculada. Las rutas que no contengan hitos constarán de un único "tramo", mientras que las rutas en las que se hayan definido uno o varios hitos constarán de uno o varios tramos correspondientes a los tramos específicos del trayecto.

Cada tramo del campo legs puede contener los siguientes campos:

- steps[] contiene un conjunto de pasos que proporciona información sobre cada uno de los pasos del tramo de un trayecto.

- distance es un campo que indica la distancia total que abarca el tramo y que consta de los siguientes elementos:

- value indica la distancia en metros.

  Text contiene una representación interpretable por humanos de la distancia, expresada en las unidades utilizadas en el origen (o modificada en el parámetro units de la solicitud). (Por ejemplo, se utilizarán millas o pies para cualquier punto de origen que se encuentre dentro de Estados Unidos). Ha de tenerse en cuenta que el campo distance.value siempre contendrá un valor expresado en metros, independientemente del sistema de unidad que se represente en el texto. Si no se conoce la distancia, es posible que estos campos no aparezcan.

- duration es un campo que indica el tiempo total necesario para recorrer el tramo y que consta de los siguientes elementos: o value indica la

duración en segundos. o text contiene una representación interpretable por humanos de la duración. Si no se conoce la duración, es posible que estos campos no aparezcan.

- start_location contiene las coordenadas de latitud/longitud del origen del tramo. Dado que el *API* de rutas utiliza la opción de transporte más cercana a los puntos de partida y de llegada (normalmente, carreteras) para calcular las rutas entre dos ubicaciones, es posible que el valorstart_locationno coincida con el origen del tramo si, por ejemplo, no hay carreteras cercanas al mismo.

- end_location contiene las coordenadas de latitud/longitud del destino dado del tramo. Al igual que el valor anterior, podría no coincidir con el final del tramo.

- start_address contiene una dirección interpretable por humanos (normalmente una calle) que refleja el valor start_location de dicho tramo.

- end_address contiene una dirección interpretable por humanos (normalmente una calle) que refleja el valor end_location de dicho tramo.

Pasos (step)

Cada etiqueta leg puede contener varios pasos steps. Cada elemento del conjunto de steps define un paso único de las rutas calculadas. Un paso es la unidad más atómica de una ruta, que consta de un único paso que describe una instrucción específica y única del trayecto. Por ejemplo, "Gira a la izquierda en la calle W. 4th St.". Un paso no solo describe una instrucción, sino que también contiene información sobre la distancia y sobre el tiempo con respecto al paso siguiente. Por ejemplo, es posible que el paso etiquetado como "Tome la interestatal 80 oeste" especifique una duración de "60 kilómetros" y de "40 minutos" para indicar que el siguiente paso se encuentra a 60 kilómetros/40 minutos.

Cada paso del campo steps puede contener los siguientes campos:

- html_instructions contiene instrucciones de formato para este paso, presentadas en forma de cadena de texto *HTML*.

- distance contiene la distancia que hay que recorrer desde un paso hasta el siguiente. Si no se conoce la distancia, es posible que este campo no esté definido.

- duration contiene el tiempo normal necesario para realizar un paso antes de pasar al siguiente. Si no se conoce la duración, es posible que este campo no esté definido.

- start_location es un conjunto único de campos de lat y de lng que indica la ubicación del punto de partida de un paso determinado.

- end_location es un conjunto único de campos de lat y de lng que indica la ubicación del punto de partida de un paso determinado.

2.1.3 Google Places API

Toda la documentación sobre esta interfaz de programación de aplicaciones ha sido tomada de la web de desarrolladores de Google [GOOGLEPLACESAPI].

Introducción

Google Places API Web Service permite consultar información sobre sitios en una variedad de categorías, como establecimientos, puntos de interés importantes, ubicaciones geográficas, etc. Puedes buscar sitios por proximidad o una cadena de texto. La búsqueda devuelve una lista de sitios junto con información resumida sobre cada sitio.

Permite realizar solicitudes de los siguientes tipos:

- *Place Searches*: permite buscar sitios dentro de un área específica. Puedes perfeccionar tu solicitud de búsqueda si proporcionas palabras clave o especificas el tipo de sitio que estás buscando.

- *Place Details*: permite solicitar más detalles acerca de un establecimiento o punto de interés determinado, solicitar más detalles acerca de un establecimiento o punto de interés determinado al iniciar una solicitud de detalles del sitio. Una solicitud de detalles del sitio devuelve información más exhaustiva acerca del sitio indicado.
- *Site Aggregate:* permite complementar los datos de la base de datos de Google Maps con los datos de tu aplicación

- *Places photos*: es una *API* de solo lectura que permite agregar contenido fotográfico de alta calidad a tu aplicación.

- *Places Autocomplete*: servicio web que devuelve predicciones de sitios en respuesta a una solicitud *HTTP*. La solicitud especifica una cadena de búsqueda textual y límites geográficos opcionales.

- *Query Autocomplete*: El servicio de autocompletado de consultas se puede usar para proporcionar una predicción de consultas para búsquedas geográficas basadas en texto al devolver consultas sugeridas mientras escribes.

Autentificación

El *API* de Google Places utiliza una clave *API* key diferente a la que utiliza *API Direcctions* y ha de ser añadida en cada solicitud que se realice y estar habilitada en la cuenta del administrador.

Búsqueda de lugares cercanos

De todos los tipos de solicitudes anteriormente comentados, para este proyecto únicamente se ha implementado la primera de ellas, Places Searches. Esta permite solicitar información sobre un lugar en las siguientes categorías: establecimientos, puntos de interés importantes y localizaciones geográficas. Se pueden buscar sitios por proximidad geográfica o a partir de un texto, y la búsqueda devuelve una lista de lugares con información básica sobre los mismos.

Una solicitud de búsqueda de sitios cercanos es una *HTTP* request como la siguiente:

```
https://maps.googleapis.com/maps/api/place/nearbysearch/output?parameters
```

donde, de nuevo, output puede ser uno de los siguientes valores: *JSON* (recomendado) y *XML*. En esta aplicación se trabaja con *JSON* (JavaScript Object Notation).

Parámetros de solicitud

Parámetros obligatorios

- *key*: la *API* key de la que se hablaba en el apartado *Autentificación*.
- *location*: El punto alrededor del cual se va a buscar, debe estar especificado por latitud y longitud.
- *radius*: Define la distancia (en metros) desde la localización a la que se realiza la búsqueda.
- *sensor*: Indica si la búsqueda ha sido realizada desde un dispositivo tipo GPS.

Parámetros opcionales

A continuación, se especifican los parámetros opcionales más relevantes.

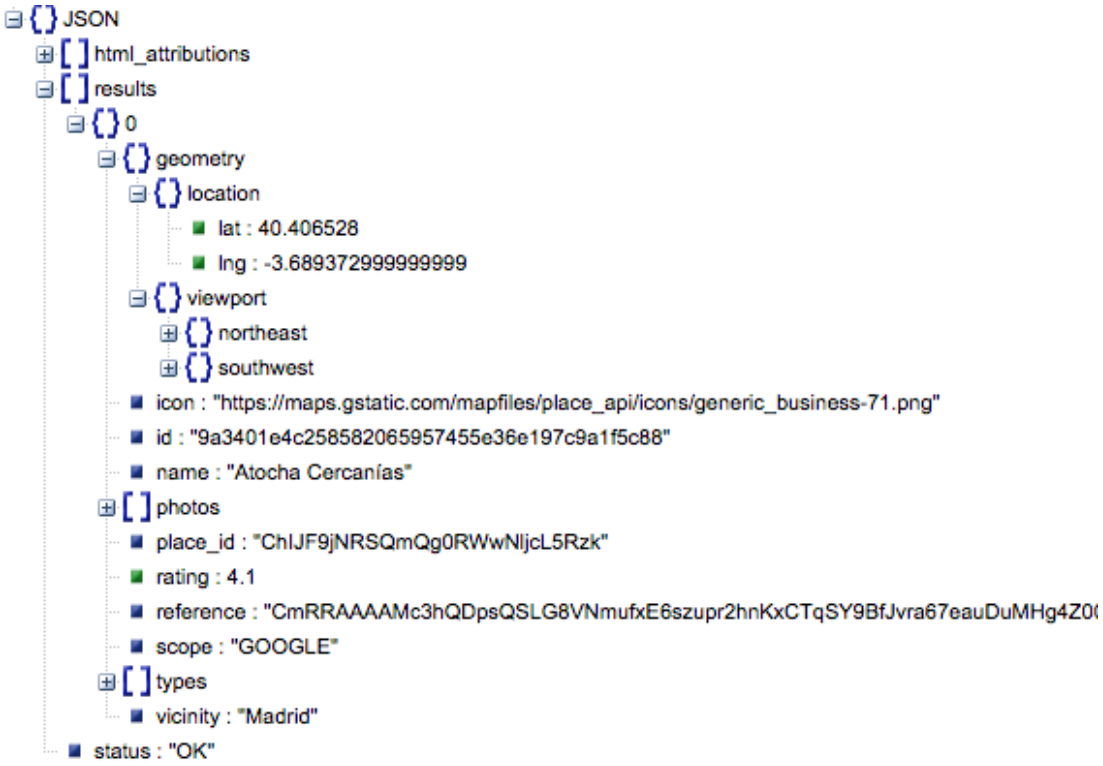
- *opennow*: devuelve solo los sitios que están abiertos en el momento en que se envía la consulta.
- *rankby:* especifica el orden en el que se indican los resultados. Este campo puede tomar los dos valores siguientes.
    1. *Prominence* (valor predeterminado). Esta opción clasifica los resultados según su importancia.
    2. *Sistance*. Esta opción clasifica los resultados en orden ascendente por su distancia desde la *location* especificada.
- *types*: restringe los resultados a sitios que coinciden con el tipo especificado.

Seguidamente se muestra un ejemplo de solicitud que permite obtener los restaurantes cercanos a la localización especificada en la solicitud entorno a un radio de 500 metros.

```
https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=-33.8670522,151.1957362&radius=500&type=restaurant&key=YOUR_API_KEY
```

Respuestas del servicio

Vienen dadas en el formato que se haya indicado en el campo output contenido en la URL de la petición. En el caso de esta aplicación es de tipo JSON.



2.1.4 Ejemplos de aplicaciones

Para el desarrollo de la aplicación del presente proyecto se ha analizado el mercado actual. **IziTravel** ha sido la única en la que me he basado para tener un punto de referencia en cuanto a nivel visual y funcional de determinados puntos.

Esta aplicación ofrece un sistema de guía turístico muy completo, con rutas seleccionables que incluyen audios informativos además de fotografías de los diferentes puntos. Cuenta con una interfaz gráfica muy cuidada con gran

funcionalidad, ya que desde cualquier actividad de la aplicación puedes acceder a cualquier vista o menú.

La intención en el proyecto fue ofrecer un conjunto mayor de funcionalidades, para ofrecer mayor potencialidad a la aplicación, cómo es la búsqueda de lugares cercanos en torno a punto, la ruta hacia ellos o la búsqueda de información en Internet sobre estos.

2.2 Big Data en combinación con Machine-Learning

2.2.1 Introducción

El aumento del volumen de datos procedentes de diferentes fuentes, como de búsquedas en la Web, redes sociales o la localización GPS entre otros, se han convertido en un activo de valor para las empresas de cualquier sector. Y es que el análisis de esta cantidad inmensa de datos está demostrado qué incrementa la rentabilidad de las empresas, aunque actualmente se estima que solo procesan el 1% de los datos que almacenan. [INVERTIA]. La necesidad de procesar estos datos ha promovido que grandes multinacionales como Google, Microsoft o IBM inviertan en el desarrollo de sistemas de inteligencia avanzada capaces de aprender de su propia experiencia. Es aquí donde la disciplina del Aprendizaje Automático (*Machine Learning*) entra en juego.

2.2.2 El valor de la información

Introducción

Lo que hace que Big Data sea tan útil para muchas empresas es el hecho de que proporciona respuestas a muchas preguntas que estas ni siquiera sabían que tenían. Esta nueva forma de utilizar los datos para identificar problemáticas o nuevas oportunidades, conlleva a las organizaciones a realizar operaciones más eficientes y rápidas, lo que se traduce en un aumento de los ingresos de estas [POWERDATA].

Ventajas

Las empresas con más éxito con Big Data consiguen valor de las siguientes formas:

Reducción de coste: gestionar un volumen tan grande de datos supone un problema a nivel de infraestructura, que las plataformas de nube resuelven. Estas permiten escalar bases de datos con facilidad, mejorar la accesibilidad y la fluidez de la información. Esto supone una reducción de costes en hardware, además de que las cuotas de tarificación de estas plataformas hacen que pagues únicamente por uso y tiempo de sus servicios.

- **Más rápido, mejor toma de decisiones**: las nuevos servicios que ofrecen las plataformas de nube permiten unificar datos de diferentes fuentes, además de transformar datos brutos en tablas para poder exportarlos en otros formatos. Esto facilitará el análisis de los datos y permitirá una toma de decisiones más rápida.

- **Nuevos productos y servicios**. finalmente el análisis de estos datos se materializan en beneficios para las empresas, ya que a partir de las tendencias o necesidades obtenidas del estudio de datos, estas crean nuevos productos o nuevas oportunidades de negocio.

Fuentes y tipos de datos

Las fuentes de datos de Big Data son muy amplias:

- Datos de internet y móviles.
- Datos de Internet de las Cosas.
- Datos sectoriales recopilados por empresas especializadas.
- Datos experimentales.

Y los tipos de datos también lo son:

- Tipos de datos no estructurados: documentos, vídeos, audios, etc.
- Tipos de datos semi-estructurados: software, hojas de cálculo, informes.
- Tipos de datos estructurados

2.2.3 Ejemplos de aplicaciones

A continuación, se van a exponer una lista de diferentes escenarios sobre los cuales se utiliza la recopilación de información para su posterior análisis y procesamiento con servicios de Machine-Learning [PWERDATA].

- **Turismo:** en este sector es clave mantener el nivel de satisfacción de los clientes elevado, y actuar antes los problemas con la mayor rapidez posible. El Big Data es la solución, este posibilita identificar de forma rápida posibles problemas y actuar para remediarlos de forma casa inmediata.

- **Cuidado de la salud**: en la actualidad en el área de la medicina es donde el Big Data está teniendo mayor impacto. Y es que este es un sector generador de una gran cantidad de datos, tanto estructurados como. Pero la calve está en estos últimos, y es que este tipo de datos procedentes de (recetas, registros, radiografías, escáneres, resonancias) poseen una información valiosa, que si se procesan de la forma adecuada contribuiría notablemente a la mejora de la investigación médica [Poyatos, s.f].

- **Administración:** si las administraciones públicas se digitalizan al igual que los servicios directos de los cuales se podrían alimentar, estas podrían ahorra costes a partir de la eficiencia de la gestión de los servicios que las ciudades pueden ofrecer a sus ciudadanos. Ya sea en, la mejora del transporte, servicios públicos, sostenibilidad energética, seguridad, etc…

- **Retail**: a través del análisis de programas de fidelización de clientes o hábitos de compra, los minoristas tienen una compresión profunda de sus clientes pudiendo predecir sus tendencias, y aumentar así su rentabilidad

- **Marketing:** los sistemas de localización de los smartphones permiten a los anunciantes la oportunidad de dirigirse a los consumidores cuando estén cerca de sus locales. Esto abre nuevos ingresos para los proveedores de servicios

- **Detección y prevención de fraudes:** en cualquier industria que procese transacciones financieras online, tales como compras, actividades bancarias, inversiones, seguros, etc. Con la ayuda del Big Data se puede mejorar los sistemas de seguridad tradicionales.

2.2.4 Plataformas para la implementación de Machine-Learning

En la actualidad el aprendizaje automático está al alcance de cualquier programador. Para experimentar estos servicios tenemos en el mercado diversas plataformas. Por lo general prácticamente todas ofrecen servicios similares y modos de uso gratuitos para poder hacer pruebas con ellos.

Destacan las siguientes:

- MeaningCloud.
- Azure Machine.
- Amazon Machine Learning

Para este proyecto se ha hecho uso de *Azure* y *MeaningCloud,* se han usado los servicios cognitivos de ambas. Con *Azure* se ha hecho uso de estos servicios en una acción de la *Logic App* para analizar los tweets de los usuarios. Por otra parte se ha usado MeaningCloud para analizar las opiniones sobre la ruta de los usuarios.

En cuanto a la integración de ambas opciones han sido diferentes:

Azure

Para hacer uso de los servicios cognitivos ha sido de forma muy automatizada. Únicamente era necesario activar este servicio desde el portal e introducir la key y el nombre del servicio en la acción de la Logic App.

MeaningCloud

El acceso a este servicio se ha hecho desde el código de la aplicación. La implementación necesaria para consumir dicha *API* ha sido igual que para el caso de las llamadas a las *APIs* de Google, con la única diferencia que en este caso la llamada era un método POST [MEANINGCLOUD]

Solicitudes de ruta

El formato de una solicitud a esta API tiene el siguiente formato.

```
https://api.meaningcloud.com/sentiment-2.1?key=&of=&lang=&ilang=&txt=
```

Parámetros obligatorios

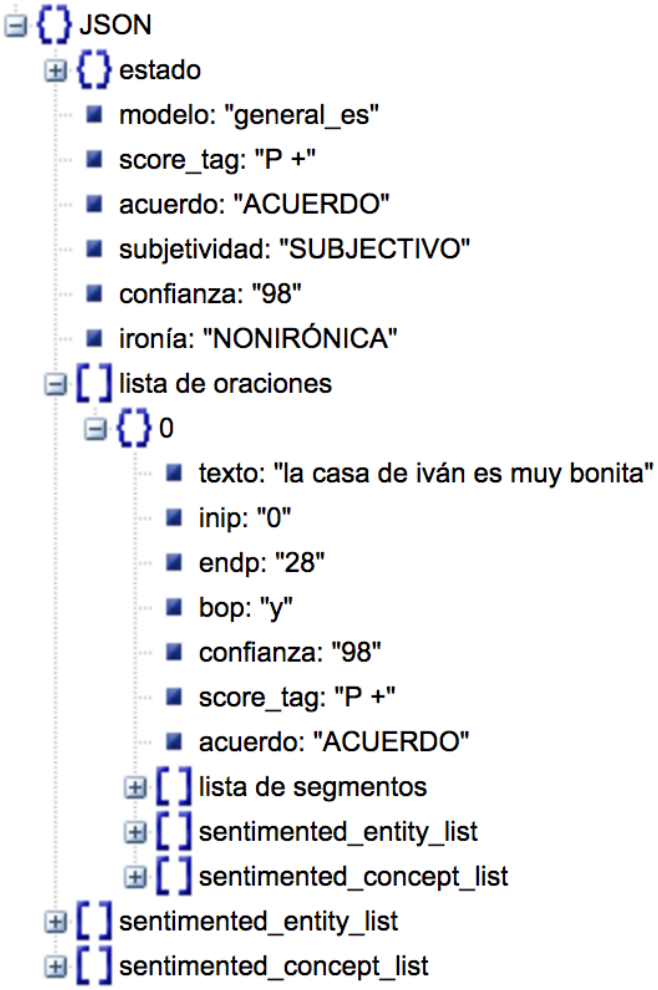
- Key: clava de acceso necesaria para realizar la solicitud.
- lang: especifica el idioma en que debe analizarse el texto.
- txt: texto a analizar por el servicio.

Parámetros opcionales

- of : se especifica el forma de salida de la respuesta del servicio, *JSON* o *XML*
- ilang: especifica el idioma en el que aparecen los valores devueltos.

Respuesta de la API

La salida contiene información sobre el estado de la solicitud, las diferentes polaridades identificadas en el texto, su subjetividad y si contiene marcas de ironía. Posteriormente se adjunta un ejemplo de una respuesta de la API.



De todos los objetos que contiene la respuesta el que nos interesa es la propiedad **score_tab** del objeto **estado**. Esta propiedad puede contener los siguientes valores:

- **P +** : positivo fuerte.
- **P** : positivo,
- **NEU** : neutro.
- **N** : negativo.
- **N +** : negativo fuerte.
- **NINGUNO**: sin sentimiento.

2.2.5 Machine-Learning vs Deep-Learning

Se puede decir que el Machine-Learning (Aprendizaje Automático) tiene una vertiente que se denomina Deep-Learning (Aprendizaje Profundo). Las dos tecnologías hacen referencia a sistemas IA capaces de aprender por sí solos, la diferencia entre ambos está en el método de aprendizaje. El de Deep-Learning es más complejo, sofisticado y autónomo, por ello una vez programado la intervención humana sobre el sistema es prácticamente nula [Bejerano, 2017].

Machine-Learning

Se proporciona a una máquina un algoritmo con un conjunto de reglas para que las aplique cuando se encuentre con los datos pertinentes. Pero el software tiene la capacidad de adaptar dichas reglas y crear otras nuevas para mejorar su tasa de acierto.

Deep-Learning

El sistema en este caso va por capas o unidades neuronales, tratando de imitar el cerebro humano. Cada capa procesa la información y arroja un resultado que se revela en forma de ponderación. Es decir, cuando una unidad de un sistema analice una foto en busca de perros concluirá que dicha imagen es en un 78% de probabilidad un perro y en un 22% no lo es.

La segunda capa que analice la imagen combinará el resultado obtenido por la primera capa con su propio juicio. De tal forma que la ponderación se modificará y se trasladará a la tercera capa , que también usará este dato para sacar su propia conclusión. Así sucesivamente y en muchas ocasiones (hay algoritmos que usan más de 100 capas).

2.2.6 Futuro hacia el Deep-Learning

Posiblemente el futuro del aprendizaje automático pase por un giro hacia el aprendizaje no supervisado. En este modelo los algoritmos son capaces de aprender sin intervención humana previa, sacando ellos mismos las conclusiones acerca de la semántica presente en los datos. Ya existen compañías que se centran completamente en enfoques de aprendizaje automático no supervisado, cuyas plataformas cognitivas son capaces de procesar millones de documentos no estructurados y construir de forma autónoma representaciones estructuradas [Arrabales, 2017]

2.3 Plataformas como servicios en la nube de Azure

Toda la información que se ha usado para la descripción de este apartado ha sido tomada de la web 'Docs' [MICROSOFTAZURE]. Es la oficial de Microsoft donde está unificada toda la documentación técnica de todos sus productos.

Introducción

Una forma de crear plataformas como servicios en la nube es mediante la creación o implementación de aplicaciones lógicas. Estas aplicaciones son una manera de simplificar e implementar flujos de trabajo, los cuales están compuestos por una serie de pasos que podemos crear a través de un diseñador visual. Una aplicación lógica comienza con un desencadenador, para después ir atravesando el conjunto de acciones de las que se componga el flujo.

Ventajas de las aplicaciones lógicas.

Entre las ventajas destacables se encuentran la velocidad y la escalabilidad que ofrecen, junto con la facilidad del diseñador y la variedad de desencadenadores o acciones que contienen.

- **Herramientas de diseño fáciles de usar**: las aplicaciones lógicas pueden diseñarse de principio a fin en el explorador o con las herramientas de Visual Studio.

- **Conexión sencilla de API**: los conectores son una parte integral de la creación de aplicaciones lógicas. Mediante el uso de conectores se puede expandir las aplicaciones locales y en la nube para realizar diferentes operaciones con los datos que cree, y con los que ya tiene.

- **Plantillas para el diseño y creación**: se dispone de una galería de plantillas que permiten crear rápidamente aplicaciones lógicas.

- **Extensibilidad incorporada**: las aplicaciones lógicas están diseñadas para funcionar con sus propias *API* y código. Puede crear fácilmente una API propia para usarla como conector personalizado o llamar a una *Azure Function* para ejecutar fragmentos de código según los necesite.

Conceptos de aplicaciones lógicas

Las siguientes son algunas de las principales partes que componen la experiencia de aplicaciones lógicas.

- **Conectores administrados**: las aplicaciones lógicas necesitan acceder a datos y servicios, esto lo consiguen a través de conectores, ya sean customizados o mismamente los propios que ya existen. Cuando un desencadenador se dispara, es necesario de un conector para seleccionar la siguiente acción en el flujo.

- **Desencadenadores**: el desencadenador inicia un nuevo subproceso en el flujo de trabajo, esta acción puede venir dada por ejemplo por la publicación de un tweet, la llegada de un fichero vía FTP, la llegada de un correo etc.

- **Acciones** : conjunto de pasos que componen el flujo de trabajo después de que se procese el desencadenador.

Conectores

Los conectores son una parte integral de las aplicaciones lógicas. Mediante el uso de conectores, puede expandir las aplicaciones locales en la nube para realizar diferentes operaciones con los datos que cree, y con los que ya tiene. Los conectores están disponibles en las siguientes categorías:

- **Conectores estándar**: automáticamente disponibles e incluidos al usar aplicaciones lógicas. Algunos ejemplos son Service Bus, DropBox, GoogleDrive, Power BI, Oracle Database o OneDrive, Facebook, Twitter, Azure Cognitives Services entre otros muchos.

- **Conectores de cuenta de integración**: están disponibles al adquirir una cuenta de integración, mediante estos conectores se puede transformar y validar código *XML*, procesar mensajes de negocio a negocio con *AS2*, *X12* o *EDIFACT*, y codificar y descodificar archivos sin formato.

- **Conectores de empresa**: incluye *MQ* y *SAP*. Suponen un costo adicional.

Para la aplicación lógica creada para este proyecto se han usado los conectores estándar, en concreto el de Twitter, Azure Cognitives Services, Azure Function y el de Slack.

Límites de uso y precios

En *Logic Apps* se miden todas las ejecuciones de acciones realizadas, esto incluye desencadenadores y acciones ejecutadas como parte de un flujo de trabajo.

A continuación, se adjunta una tabla donde están reflejados los precios actuales por número de acciones.

Acciones de Logic Apps

ACCIONES EJECUTADAS/MES	EJECUCIÓN DE PRECIO POR ACCIÓN
Primeras 250 mil acciones	€0,000675/acción
Entre 250 mil y 1 millones de acciones	€0,000338/acción
Entre 1 y 50 millones de acciones	€0,000127/acción
Entre 50 y 100 millones de acciones	€0,000076/acción
Más de 100 millones de acciones	€0,000046/acción

Figura 5: Precios acciones Logic Apps

Ejemplos de tipos de escenarios

Seguidamente se exponen dos escenarios con diferentes flujos de trabajo en los que se combinan *Azure Functions* y *Logic Apps* [MICROSOFTAZURE].

Crear un panel social sin servidor

Las herramientas de *Azure* sin servidor proporcionan funcionalidades eficaces para generar rápidamente y hospedar aplicaciones en la nube, sin tener que preocuparse de la infraestructura. En este escenario, se creará un panel para actuar en función de los comentarios de los clientes, analizarlos con Machine Learning y publicar información de un origen como Power BI o Azure Data Lake.

Llamadas a aplicaciones lógicas con Azure Functions

En este otro escenarios se usan Funciones de Azure para crear un desencadenador para una aplicación lógica cuando necesite implementar un agente de escucha o una tarea de ejecución prolongada. Por ejemplo, puede crear una función que escuche en una cola y que active inmediatamente una aplicación lógica como desencadenador de push.

Capítulo 3

DESCRIPCIÓN DE LA APLICACIÓN DESARROLLADA

Una vez analizado el estado del arte en el que se enmarca esta aplicación, se describen en este capítulo las características fundamentales del sistema desarrollado: conjunto de funcionalidades, tecnologías implicadas y esquema general.

Para ello se comenzará con una visión general del proyecto para profundizar posteriormente con descripciones más detalladas de cada punto.

3.1 Presentación de la aplicación

La aplicación desarrollada para el Trabajo Fin de grado consiste en un asistente turístico que ofrece los siguientes servicios:

- Búsqueda de rutas a través de una selección de opciones mediante filtros y listado de las rutas encontradas.
- Resumen de la ruta preelegida con las diferentes fotos de los puntos que la componen. Además de las opiniones de los usuarios.
- Representación en el mapa de la ruta con los diferentes puntos de los cuales está compuesta.
- Reproducción de audio con información turística de cada uno de los puntos de la ruta. También se añade la posibilidad de leer dicha información.
- Guardado dinámico de los puntos que el usuario va completando.
- Opción de búsqueda de lugares de interés cercanos entorno un punto elegido de la ruta.
- Búsqueda en Internet información sobre un sitio de interés elegido.
- Calcular la ruta a un sitio cercano que el usuario haya elegido.

- Posibilidad al usuario de añadir la opinión de la ruta de forma oral y análisis de esta.
- Conjunto de tres tablas de una base de datos SQLite para almacenar la información, tanto del usuario como la de la propia aplicación.

La aplicación da al usuario un servicio completo, que abarca desde una una selección de rutas mediante filtros, pasando por un asistente turístico mediante la reproducción de audios. Llegando a dar la posibilidad de buscar sitios de interés cercanos y calcular la ruta hacia a ellos.

En total la aplicación facilita 5 funcionalidades fundamentales. A continuación, se describen con mayor detalle los módulos de la aplicación y las actividades que se pueden realizar.

3.2 Búsqueda y listado de rutas

Selección de filtros

Esta funcionalidad queda recogida en la primera actividad de la aplicación, Figura 9. En ella vemos un conjunto de elementos *XML* conocidos como Spinners, estos funcionan a modo de filtros y permiten al usuario elegir una ruta de acuerdo a unas preferencias. De los cuatro filtros que aparecen, sólo son obligatorios los tres primeros. La opción de la nota es opcional, en el caso de que el usuario no envíe nada en ese campo la aplicación devolverá todas las rutas que cumplan el resto de opciones enviadas.

Búsqueda de las rutas en base de datos

Una vez que el usuario selecciona sus preferencias de búsqueda y pulsa el botón *Búscar*, el flujo que se desencadena en la aplicación es el siguiente:

- Se inicia el cambio a la nueva actividad. Al objeto *Intent* que carga la nueva actividad se le pasan los valores que se ven en la Figura 6.

```
intent.putExtra("CITY", preferencesOptionsUserCity);
intent.putExtra("THEME", preferencesOptionsUserTheme);
intent.putExtra("DISTANCE", preferencesOptionsUserDistance);
intent.putExtra("NOTE_ROUTE", preferencesOptionsUserNote);
```

Figura 6: Datos que se pasan de la Actividad 1 a la 2

- Ya en la segunda Actividad, Figura 10, recogemos los valores que nos llegan de la primera Actividad y se inicia la búsqueda en base de datos de la rutas.

- La query que se realiza en este caso sobre la tabla *routes.sqlite,* es un SELECT cuya cláusula WHERE necesita tres valores *typeRoute, city, distance, opinion*. Debajo, en la Figura 7, se ve una entrada de la tabla *routes.sqlite.* El resultado de la búsqueda se almacena en una variable de tipo Cursor, que contiene una lista de rutas. Donde para cada ruta se guarda las siguientes propiedades *name*, *summary*, *coordinates*, *query*, *numpoints* y *namepoints*. Más adelante se explica cuándo y para qué se usa cada uno de los valores.

_id	name	summary	coordinates	description	opinion	query	typeRoute	npoints	city	distance	namespoin...
1	Tour Cultural	A través de esta ruta podrás ver l...	40.407635,-3.6...	Madrid es ...	2	https://ma...	Cultural	6	Madrid	Media	Reina Sofía,At

Figura 7: Base de datos routes.sqlite

- Por último, la lista de rutas encontradas se carga en un elemento ListView para poder listarlas y que el usuario elija, como se ve en Figura 10.

Una vez que el usuario ha elegido una ruta del listado, se procede el cargado de la siguiente Actividad. El *Intent* que carga la actividad se le pasan los valores que se ven en la Figura 8, estos son los obtenidos en la anterior query, pero de la lista de rutas que teníamos ahora nos quedamos con la elegida por el usuario.

```
intent.putExtra("NANE_ROUTE", cursor.getString(1));
intent.putExtra("COORDINATES",cursor.getString(3));
intent.putExtra("QUERY", cursor.getString(4));
intent.putExtra("POINTS", cursor.getInt(5));
intent.putExtra("NAMEPOINTS", cursor.getString(6));
intent.putExtra("USER", user);
intent.putExtra("CITY",nameCity);
```

Figura 8: Datos que se pasan de la Actividad 2 a la 3

A continuación, vemos las vista de las dos primeras actividades que recogen las funcionalidades anteriormente descritas.

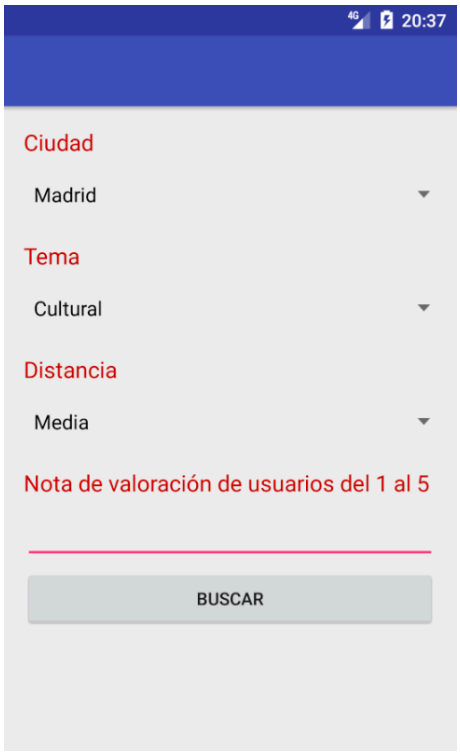


Figura 9: Pantalla de inicio, filtros de búsqueda, Actividad 1

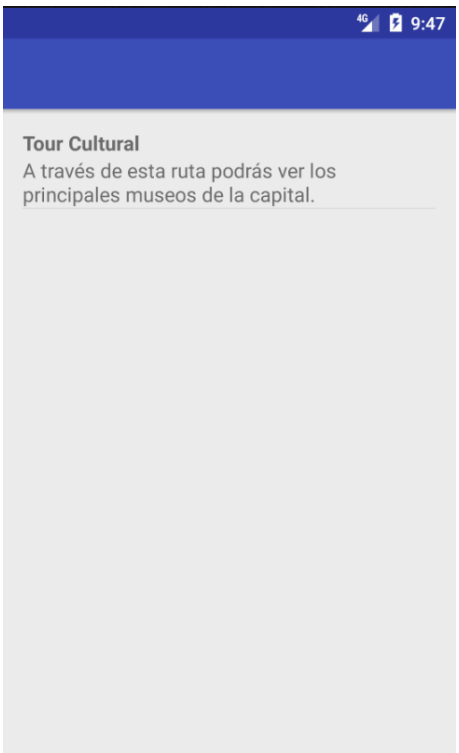


Figura 10: Listado de rutas, Actividad 2

Añadir un pequeño inciso, el parámetro *User* no se obtiene de la base de datos, es un valor fijado en la aplicación, ya que en un principio se pensó añadir un registro de usuarios, pero al final no dio tiempo. Aunque la implementación de la aplicación está preparado para soportarlo.

3.3 Imágenes, resumen y opiniones de la ruta preelegida

Una vez hemos elegido de la posible lista de rutas una de ellas, avanzamos a la siguiente Actividad, Figura 11. Esta Actividad está formada por tres partes, que aportan diferente información de la ruta al usuario.

Imágenes de la ruta y título

Esta parte está formada por los siguientes elementos XML, cuyo aspecto visual se puede ver en la Figura 11.

- ImagenButton que permite al usuario pasar las fotos.
- ImagenSwitcher es el elemento en el cual mostramos las diferentes fotografías de los puntos de la ruta.
- TextView que contiene el título de la fotografía, que cambia dinámicamente según se van pasando las fotos.

Las fotografías que se muestran se encuentran guardadas dentro de la aplicación, para cargarlas en necesario de una clase auxiliar, donde a un método estático se le pasa el nombre de la ruta elegida, y a través de un Switch devuelve un Array que contiene los nombres de las fotografías para esta ruta.

Resumen de la ruta

A nivel de diseño esta parte de la vista únicamente contiene un ScrollView en combinación con un TextView. El primer elemento es necesario en el caso que el texto a mostrar sea más largo de lo que se vea por defecto en el hueco de la pantalla. Y el segundo elemento únicamente muestra el texto.

El contenido del texto que se muestra en la Figura 11, es a modo de resumen informativo sobre la ruta, este se obtiene de la base de datos. La información es el valor de la columna *summary,* Figura 7. Para obtener la información realizamos un SELECT en nuestra tabla *routes.sqlite* cuyas cláusulas WHERE son nombre de ruta y ciudad. Estos valores había sido pasados desde la Actividad anterior, tal y como se ve en la Figura 8.

Opiniones de los usuarios

Lo último que el usuario ve en esta ruta será el listado de las opiniones de los usuarios que ya han recorrido esta ruta. Para mostrar esta información volvemos a usar un ListView, el usuario podrá hacer scroll para ver todas las opiniones seguidas cada una de su nota.

De nuevo para este caso, volvemos a usar una base de datos para obtener todo el listado de opiniones, esta información la obtenemos realizando una búsqueda en la tabla *pointsTracked.sqlite*. El esquema de esta tabla se ven en la Figura 17, y la query que se realizar para obtener el listado es un SELECT cuyas cláusulas WHERE es el nombre de la ruta. El aspecto de esta parte de la Actividad queda reflejado en la Figura 11.

Las siguientes imágenes representa la vista de la tercera Actividad de la aplicación.

Figura 11: Resumen de ruta preelegida

3.4 Representación en el mapa de la ruta y reproducción de audios

Una vez que el usuario ha pulsado al botón de *iniciar ruta*, llegamos a la cuarta Actividad, Figura 12, 13 y 14, está implementada la mayor parte de la funcionalidad. La información que nos llega de la Actividad anterior, es la misma que recibía la Actividad tres y la cual está reflejada en la Figura 8.

Posteriormente se explica todas las posibilidades que se le da al usuario mientras se mantenga en esta Actividad, y cómo están implementadas cada una de ellas, se explicarán siguiendo el orden decreciente del nivel de funcionalidad principal a secundaria u opcional.

Representación de la ruta en el mapa

Una vez nos encontramos en esta Actividad, el usuario ve representado en el mapa en azul los diferentes puntos y el trayecto de la ruta elegida. Para poder representar dicha ruta se ha hecho uso de la *API Google Directions*.

A partir de la variable *Query*, que obtenemos de la Actividad anterior, Figura 8, que almacena la url de la request, el programa hace una petición de ruta mediante la API mencionada, esta devuelve la ruta en un elemento HashMap.

HashMap es una colección de objetos de mapa, cada uno identificado por una clave. Estos resultados se guardan en el programa en una lista de puntos de tipo LatLng (coordenadas de latitud-longitud) y a continuación se unen los puntos mediante un objeto de tipo PolylineOptions, el cual permite añadir una línea con las características definidas por el desarrollador (color azul, grosor de 7 píxeles, etc.) que une dos o más puntos.

La implementación de esta funcionalidad se ha realizado de forma asíncrona, ya que el tiempo de espera mientras se obtiene la respuesta de la API puede provocar que la aplicación se cierre.

Figura 12: Representación de la ruta, Actividad 4

Figura 13: Barra inferior tras seleccionar punto

Reproducción de audios cuando se elija un punto

Una vez que el usuario seleccione un punto de la ruta, aparecerá en la parte inferior un cuadro con diferentes botones e información, tal y como se ve en la Figura 13.

En el caso de que el usuario haga click en el botón *Play*, la aplicación empieza a reproducir un audio con la información turística del punto que haya elegido. Para poder añadir esta funcionalidad, hemos usado una instancia de tipo MediaPlayer, esta clase permite controlar la reproducción de archivos y secuencias de audio / vídeo. El usuario puede pausar el audio y reanudarlo cuando desee, y en el caso de que seleccione otro punto diferente el audio se detendrá automáticamente para que se pueda reproducir el nuevo audio.

A parte de la reproducción del audio, en la pantalla aparece una barra de progreso que cuantifica el tiempo del audio que ha pasado. De nuevo esta funcionalidad se ha implementado de forma asíncrona, y esta vez era necesario, ya que si no se hacía de tal forma la pantalla queda congelada mientras avanza la barra de progreso . En la siguiente imagen, Figura 14, se ve la barra en pleno progreso.

Figura 14: Barra de progreso en curso tras iniciar la reproducción del audio

Figura 15: Información adicional del punto de la ruta seleccionado

En cuanto el almacenamiento de los audios cabe comentar que, al igual que las fotografías, estos se han almacenado en el interior de la aplicación. El formato en el que han sido guardado es *Ogg,* que permite crear archivos de audio en un tamaño muy reducido, para poder conseguir archivos de audio en este formato he usado *Switch*, que da la posibilidad de convertir audios *Mp3* a *Ogg*.

Información adicional del punto seleccionado

Una vez que hemos seleccionado un punto de nuestra ruta, aparecen un conjunto de botones, como se ven en la Figura 13 o 14, donde el botón *Info* da la posibilidad al usuario de poder leer la información en lugar de reproducir el audio, o ambas cosas a la vez, Figura 15.

Esta información se obtiene de la base de datos, la estructura de esta se ve en la Figura 16. Únicamente se usa para obtener la descripción del punto que actualmente se encuentra seleccionado, a través de una consulta SELECT en la tabla *NamePointText.sqlite* con una clausula WHERE del nombre del punto cuyo valor nos llegó de la Actividad anterior, Figura 8.

id	namePoint	description
1	Reina Sofía	museo reina sofía
2	Atocha Estación	es la estación de atocha

Figura 16: Base de datos para obtener la descripción del punto.

En la Figura 15 se ve el aspecto de la pantalla del terminal cuando el usuario ha hecho uso de esta funcionalidad.

Almacenamiento de los puntos recorridos

Otra de las funcionalidades que ofrece la aplicación es un guardado en base de datos de los puntos que el usuario va recorriendo, cada vez que el usuario selecciona un nuevo punto de la ruta que no ha ya seleccionado antes, se produce una actualización en base de datos de los puntos que lleva recorridos.

La base de datos que se ha usado para este caso tiene la estructura de la Figura 17, y la actualización se produce en la columna *pointTracked,* se van añadiendo en tiempo real los valores de los puntos que se van seleccionando.

_id	nombreRuta	nameUser	pointTranked	score	opinion
1	Tour Cultural	Ivan	0de6,	1	1

Figura 17: Tabla donde se almacena información de usuario

El usuario podrá acceder a esta funcionalidad haciendo click en el botón del menú superior que aparece en la Figura 19. Inicialmente tiene el valor "0de6", donde la primera cifra, que es la que permanentemente ve el usuario, va aumentando según vaya completando puntos. Con esta función se espera mejorar la usabilidad de la aplicación, dado que en cualquier momento el usuario puede ver el histórico de los puntos recorridos. El aspecto gráfico de esta funcionalidad se aprecia en la Figura 18. La información que se muestra es el número que tiene asignado dicho punto, y su nombre.

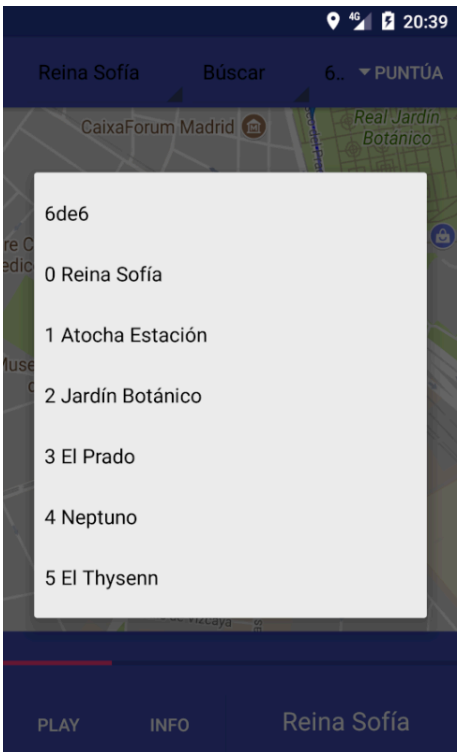


Figura 18: Histórico de los puntos recorridos por el usuario

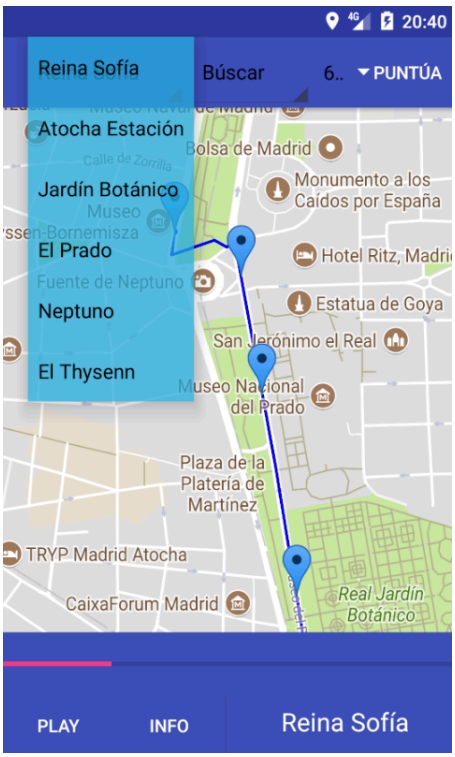


Figura 19: Listado del nombre de todos los puntos de la ruta

Listado de los nombres de los puntos de la ruta

En el menú superior de esta Actividad, tal y como se ve en la Figura 19, tenemos un desplegable, cuando el usuario hace click en él se ven todos los puntos que componen esta ruta. Esta función se añade para mejorar la experiencia de usuario, para que pueda ver en cualquier momento de una forma rápida el listado del conjunto de puntos que compone la ruta.

3.5 Búsqueda de lugares cercanos y cálculo de ruta

En este apartado se van a describir dos funcionalidades extras que complementan a las que ya hemos visto, ofreciendo así al usuario un rango más amplio de funcionalidades. No solo nos limitamos al ámbito turístico, sino a las necesidades que le puedan surgir a un usuario durante el uso de la aplicación, necesidades como localizar un lugar cercano al punto en el que se encuentra, lugares como una parada de metro, restaurante, hospital, etc.

Búsqueda de lugares cercanos de interés

Para que la aplicación muestre los lugares cercanos es necesario que el usuario haya seleccionado un punto de la ruta. A través del menú de la barra superior, Figura 19, podemos acceder a esta funcionalidad clicando en el botón *Búscar*.

Una vez que el usuario selecciona esta opción, se despliega un listado de los tipo de lugares sobre los que la aplicación es capaz de realizar la búsqueda. En la Figura 20 se puede ver dicho listado

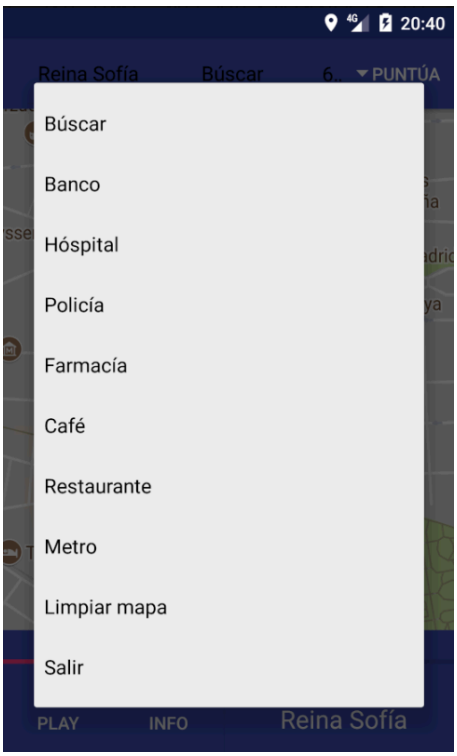


Figura 20: Listado de lugares cercanos posibles

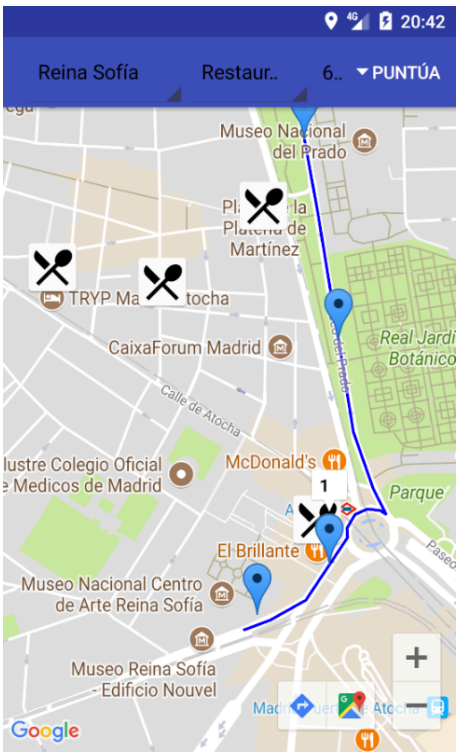


Figura 21: Representación de los restaurantes cercanos entorno al punto 1

Tras seleccionar un lugar, enviamos una request para consumir Google Places API, esta API nos devolverá los puntos cercanos del tipo seleccionado, una vez mapeado la response de la API, obtenemos una lista de HashMap, donde cada elemento de la lista es un lugar cercano para el que se han mapeado el nombre de lugar y la posición.

Tras realizar la request y el mapeo, añadimos los Markers en el mapa, dependiendo del tipo del lugar cercano la imagen del Marker que lo representa cambia. En la Figura 21 se puede ver cómo quedan representados los puntos en el mapa cuando se han buscado restaurantes.

Si el usuario decide buscar otro tipo de lugar cercano, la aplicación elimina los que había y representa los nuevos. A parte, entre los diferentes tipos de lugares que se pueden buscar, en el selector tenemos la opción de limpiar el mapa, esto es útil a la hora de poder eliminar los puntos de interés cercanos que haya en ese momento.

Búsqueda en Internet del lugar seleccionado

Cuando tenemos los lugares cercanos representados en el mapa, el usuario tiene la opción de pinchar sobre él y elegir entre dos opciones que son informadas mediante un cuadro de diálogo, como se ve en la Figura 22.

Si el usuario selecciona la opción de buscar en Internet, a través de un Intent se abre el navegador nativo del Smartphone con una pestaña, que muestra el resultado de la búsqueda en Google con el nombre del lugar seleccionado, como se aprecia en la Figura 23

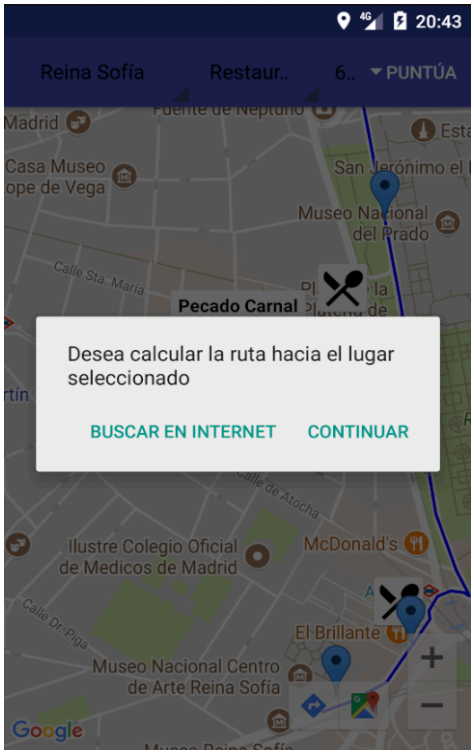


Figura 22: Cuadro de diálogo al seleccionar un lugar cercano

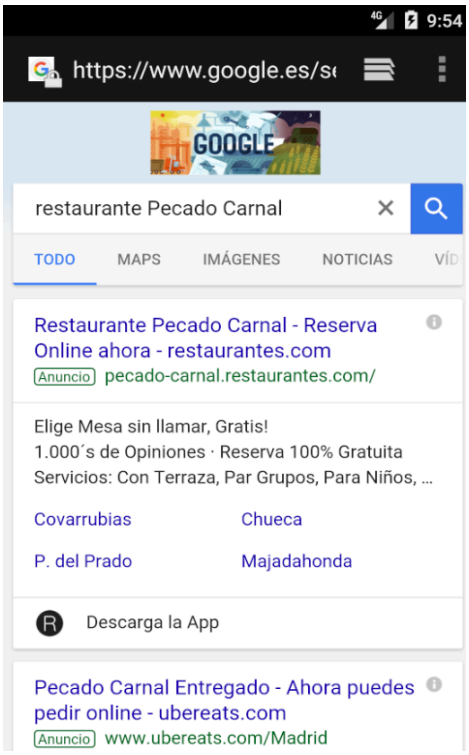


Figura 23: Resultado de la búsqueda en internet del lugar seleccionado

Ruta hacia el lugar cercano seleccionado

En caso de que el usuario elija la opción de calcular la ruta hacía el punto seleccionado en el cuadro de diálogo anterior, la aplicación reaccionará de la siguiente forma.

- Se limpiarán todos los elementos que hayan representados en el mapa en ese momento.
- Enviará un request para consumir de nuevo la *API Direcctions*.
- Tras mapear la response de la *API*, se representará en el mapa la ruta. El punto de origen será el punto de la ruta en el que se encuentre y el de destino el punto de interés que haya seleccionado.

Si el usuario desea volver a ver la ruta turística representada en el mapa, se le dará la opción haciendo click en el menú superior en el botón *Ruta*. En la Figura 24 se ve como queda representada la ruta en el mapa, y el botón para volver a la ruta inicial.

Figura 24: Representación de la ruta hacia el lugar cercano seleccionado

Durante este proceso, la opción de buscar lugares cercanos sigue estando activa, si el usuario decide buscar otro lugar cercano y calcular la ruta hacía él o buscarlo en Internet podrá hacerlo tal y como se ha explicado en el anterior apartado.

3.6 Análisis de sentimientos y guardado de la opinión del usuario

Una vez que el usuario haya completado todos los puntos de la ruta podrá comentar su opinión para valorarla. Para llegar a esta Actividad, Figura 25, se hará clicando en el botón *Puntúa* de la Figura 21, una vez que el usuario se encuentra en esta Actividad pulsando el botón principal podrá decir oralmente su opinión.

Cuando el usuario haya dicho su opinión, la aplicación llevará a cabo un flujo en el cuál enviará una request *POST* a un servicio para el análisis de sentimiento, y una vez obtengamos la respuesta de la *API* actualizaremos diferente información en dos tablas distintas.

1. En primer lugar, se actualizará las columnas *Opinion* y Score de la tabla *PointTracked.sqlite.* En el primero de ellos se guarda en formato texto la opinión del usuario, y en el segundo el valor de la puntuación ofrecida por el servicio en base a su opinión.

2. La otra tabla que se actualiza es la columna de Opinion de *routes.sqlite.* En ella se almacena el valor de la media de la opinión de todos los usuarios que han recorrido esa ruta.

Este proceso se lleva a cabo cada vez que un usuario completa una ruta, de tal forma que, cuando un nuevo usuario seleccione esa ruta en la Actividad del resumen de la ruta pre-elegida podrá ver las opiniones, y la nota de los anteriores usuarios.

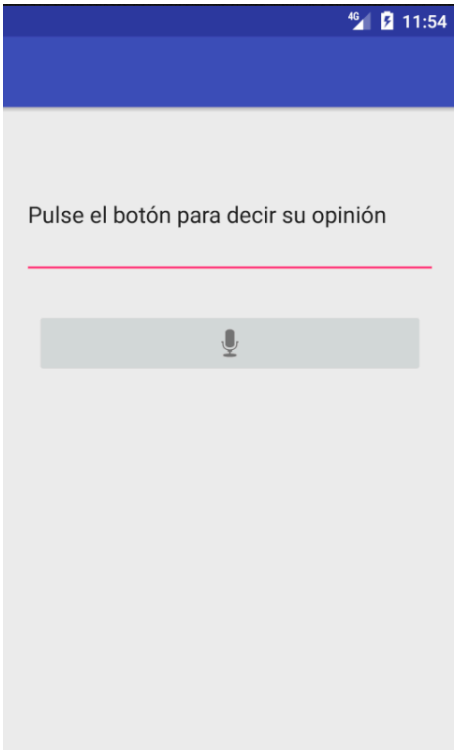


Figura 25: Análisis de opinión del usuario

3.7 Tablas de bases de datos utilizadas

Para el desarrollo de la aplicación y como ya se ha visto en los puntos anteriores, han sido necesario la implementación de tres tablas de bases de datos de tipo *sqlite*. Estas están almacenadas en el interior de la aplicación y para la gestión del contenido de las tablas me he ayudado a través *SqliteManager*, es un complemento de Firefox con el que se puede manipular de forma fácil tablas de tipo *sqlite*.

En primer lugar, la tabla que forma la columna vertebral de la aplicación *routes.sqlite*, Figura 7. A continuación se detalla brevemente para que se usa cada uno de sus campos:

- Los campos Name, Summary, City, Distance, TypeRoute son de tipo *text*, y principalmente se utilizan para realizar búsquedas en base de datos que se llevan a cabo durante la ejecución de la aplicación. Como a la hora de que el usuario envíe sus preferencias de búsqueda en la Actividad 1, Figura 6.

- El campo Coordinates, de tipo *text*, se usa para añadir cada uno de los puntos de la ruta en el mapa.

- La propiedad Query, de tipo *text*, se guarda la request que se envía a la *API Direcctions* para que nos devuelva la ruta.

- En la columna Opinion, de tipo *text*, se guarda la nota de la ruta, esta se calcula a partir de las respuestas recibidas tras el análisis de sentimientos sobre las opiniones de los usuarios.

- El campo N_points, de tipo *Integer*, se usa como comprobador y saber cuándo un usuario ha completado todos los puntos de la ruta. Para que este pueda puntuarla.

- Y, por último, la columna NamePoints, en ella se almacena el nombre de cada uno de los lugares que componen la ruta. Para poder mostrarlos en las partes de la aplicación donde se demande. Por ejemplo, para la Figura 19.

La segunda tabla que usamos es *PointTracked.sqlite*, Figura 17, la usamos para guardar información del usuario:

- En la columna pointTracked de tipo *text*, se irán guardando de forma dinámica los puntos que vaya recorriendo el usuario.

- En el campo Score de tipo *Integer*, se guarda el mapeo de la nota que nos devuelva el servicio de análisis de sentimientos, una vez enviada la opinión del usuario. Con este valor se hace una media con las notas de los usuarios que haya recorrido la ruta, y se actualiza el Score global de la ruta en la tabla *routes.sqlite*. Esta nota será por la que luego puedas filtrar en la Actividad 1, Figura 9.

- Y en el campo Opinion de tipo *text,* se guarda el texto de la opinión que haya dado el usuario. Necesario en la Actividad 2, Figura 11, a la hora de listar las opiniones de los usuarios sobre la ruta seleccionada.

Y la última tabla utilizada, Figura 16, simplemente obtenemos del texto con el contenido equivalente al del audio, para que el usuario pueda leer en lugar de escuchar la información turística del lugar seleccionado.

3.8 Logic App en combinación con Azure Functions

En este apartado se va describir el funcionamiento de la plataforma creada en la nube de *Azure* que provee de un servicio para procesos de negocio de forma automatizada. Esta plataforma se basa en una *Logic App* (aplicación lógica) que se ejecuta cada un tiempo x configurado. Tras ejecutarse procesa los tweets para finalmente publicarlos junto con su firma en el canal de *Slack* correspondiente (malo, bueno o muy bueno), según el análisis de sentimientos del contenido del tweet.

Diagrama del Flujo de trabajo

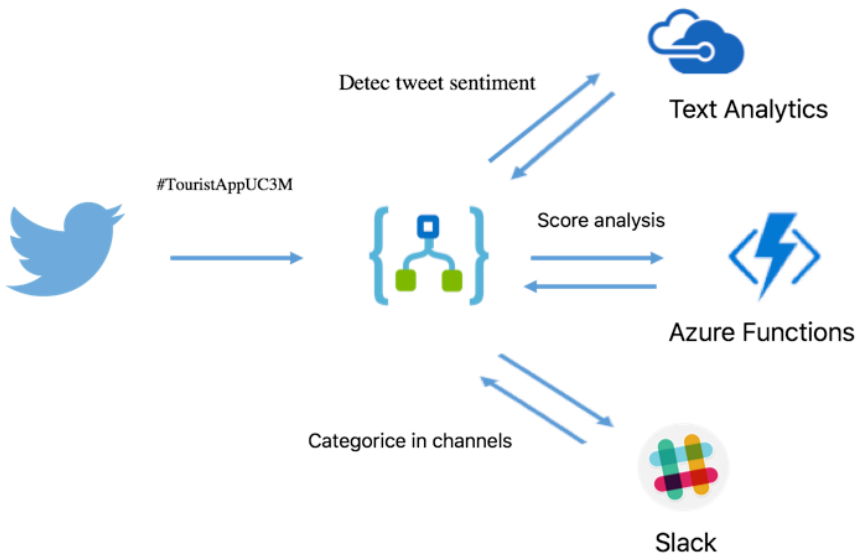


Figura 26: Diagrama de flujo Logic App

Creación y configuración de las acciones de la Logic App

Seguidamente se van a describir cada una de las acciones que conforman el flujo de la *Logic App*.

- **Acción desencadenadora**: desde el portal de *Azure* se dispone de un diseñador para crear de forma intuitiva la *Logic App*. En primer lugar, elegimos la acción desencadenadora del flujo, esta acción se ejecuta cada 3 minutos, y analiza cada uno de los tweet del tablón de la cuenta, y aquellos que contengan el #TouristAppUC3M serán enviados a la siguiente acción. En la siguiente imagen se muestra una captura del aspecto de esta acción en el diseñador.

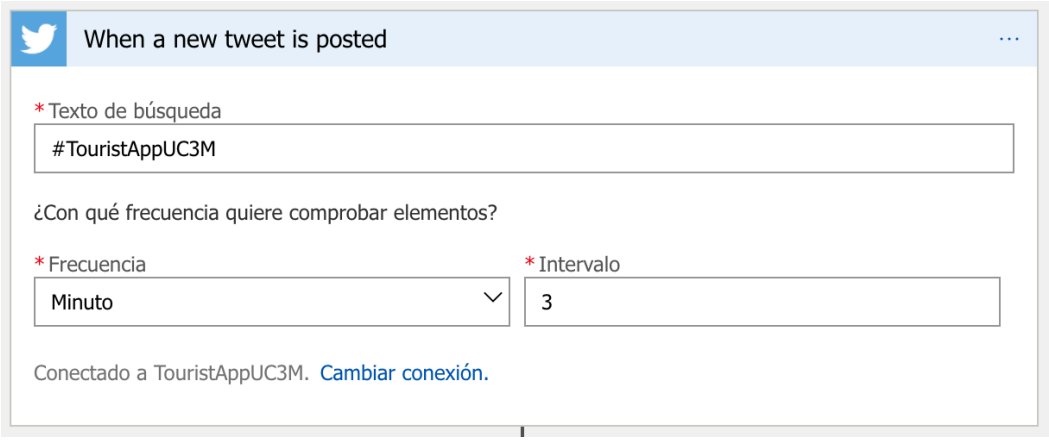


Figura 27: Acción desencadenadora Lógico App

- **Acción análisis de sentimientos**: en la segunda acción se manda el contenido del tweet a la API de *Azure* de Cognitivo Servicies, previamente ese servicio ha sido activado para poder usarse. El resultado devuelto por la API se pasará a la siguiente acción, este será un número entre 0 y 1, en la siguiente captura de muestra dicha acción.

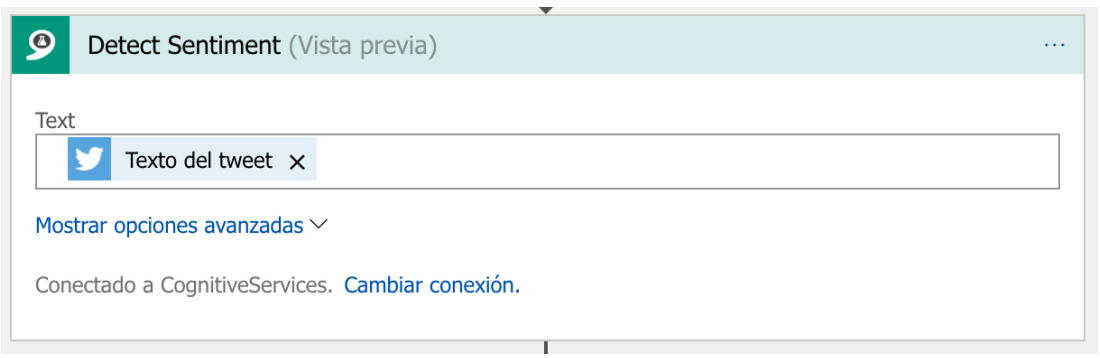


Figura 28: Acción análisis de sentimien

- **Azure Function:** en esta tercera acción se envía a la función la propiedad JSON "score" de la respuesta de la API. La función retorna un texto (Malo, Bueno o Muy bueno) según el valor del score. Esta función ha sido configurada y creada previamente en lenguaje C#. La siguiente imagen muestra dicha acción.

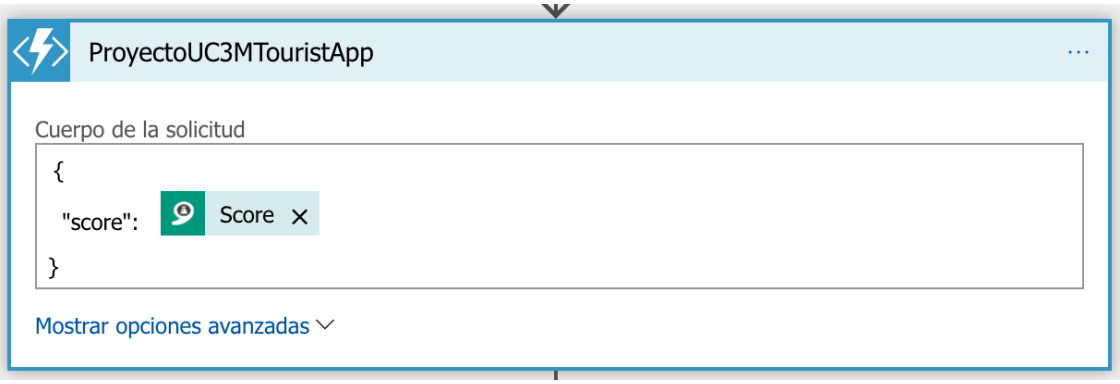


Figura 29: Acción Azure Function

- **Conmutador:** la última acción está compuesta por un conmutador con tres casos, a este se le pasa el cuerpo de la respuesta de la función. Este conmutador está conectado a una cuenta de Slack con tres canales de difusión (Malo, Bueno y Muy bueno), y dependiendo del cuerpo de la respuesta el conmutador publicará el nombre del usuario y el texto del tweet en un canal u otro. En la siguiente imagen se muestra una condición del conmutador.

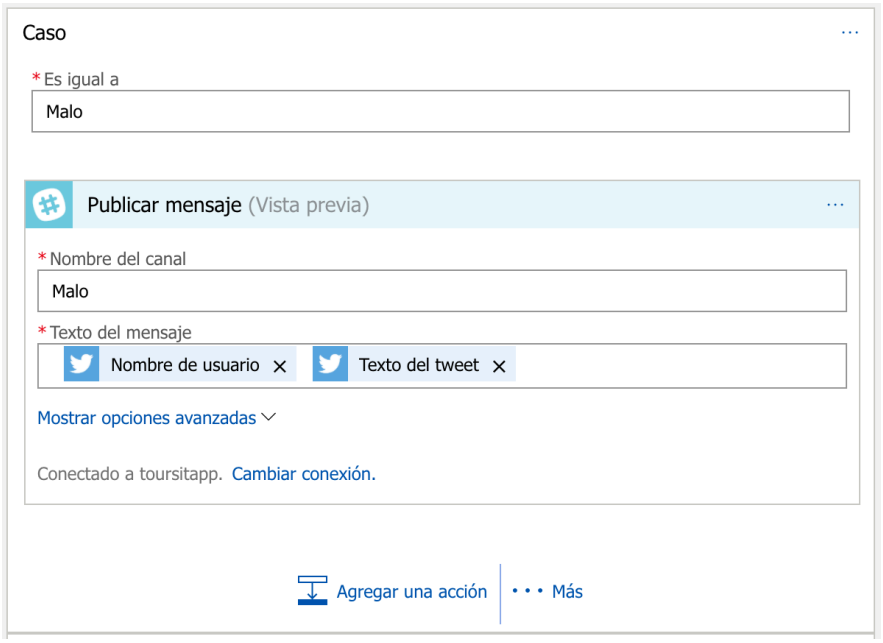


Figura 30: Conmutador Logic App

Creación y configuración de la Azure Function

Las *Azure Functions* es una buena solución para la implementación de pequeños fragmentos de código que realizan una función muy específica y no requieren de una infraestructura para ejecutarla. Soportan además diferentes lenguajes de programación C#,C, Node.js, Python o PHP.

Antes de implementar el código de la función es necesario una configuración previa.

- Tipos de métodos http de entrada permitidos.
- Nombre del parámetro de entrada.
- Nivel de autorización.

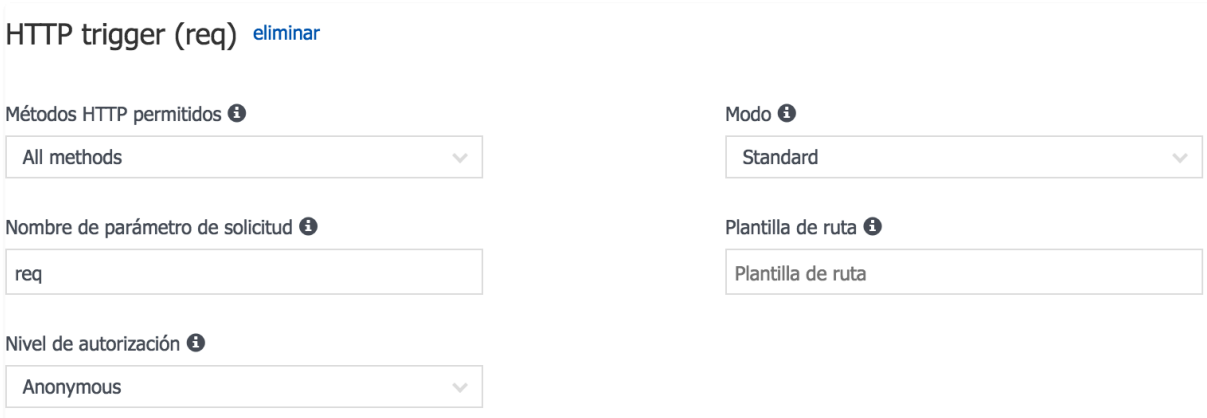


Figura 31: Configuración Azure Function

El código de la función es muy simple, recibe una request de entrada que serializa a formato *JSON* para obtener el parámetro score. En función del score un conjunto de if anidados retornan un string, para que finalmente la función retorne una respuesta http con el contenido del string.

```
#r "Newtonsoft.Json"
using System.Net;
using System.Threading.Tasks;
using System;
using Newtonsoft.Json;

public static async Task<HttpResponseMessage> Run(HttpRequestMessage req, TraceWriter log)
{

    string json = await req.Content.ReadAsStringAsync();

    dynamic data = JsonConvert.DeserializeObject(json);

    double score = data.score;

    string sentiment = string.Empty;

        if(score < 0.2)
    {
        sentiment = "Malo";
    }
    else if (score < 0.7)
    {
        sentiment = "Bueno";
    }
    else
    {
        sentiment = "Muy bueno";
    }

    return sentiment == null
        ? req.CreateResponse(HttpStatusCode.BadRequest, "Please pass a name on the query
        string or in the request body")
        : req.CreateResponse(HttpStatusCode.OK,sentiment);
}
```

Figura 32: Función Azure Function

Creación y configuración o conexión para el uso de Cognitive Services

A la hora de configurar y crear la *API* de *Cognitive Services* de *Azure* simplemente hay buscar el recurso dentro del Portal y activarlo. Únicamente hay que elegir el tipo de API con la que queremos trabajar, en mi caso es Text Analytics API, una vez hecho esto, para añadirlo al flujo de la *Logic App* solo hay que introducir el nombre de la conexión y la key.

Servicios de trazabilidad y otros que nos ofrece Azure

Por lo general *Azure* ofrece servicios de trazabilidad sobre todos sus recurso, esto nos permite identificar anomalías o errores de forma rápida y precisa. A parte de estos servicios, *Azure* nos ofrece otros que se indicarán en a continuación.

- En el recurso de la *Logic App* disponemos de un menú que nos ofrece una gran cantidad de opciones. Entre ellas cabe destacar las siguientes.

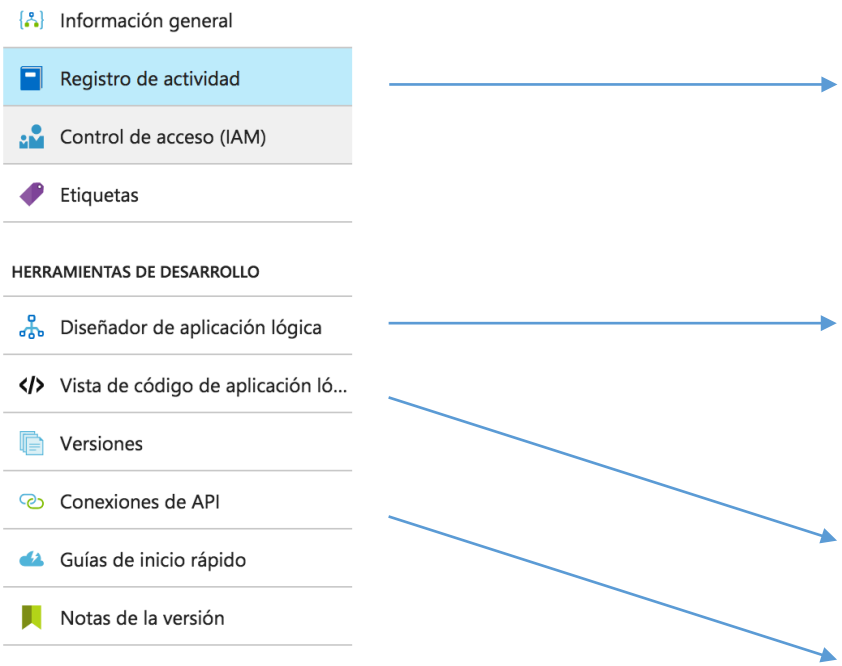


Figura 33: Menú Logic App

- En las *Azure Functions* disponemos de un menú de opciones parecido al anterior.

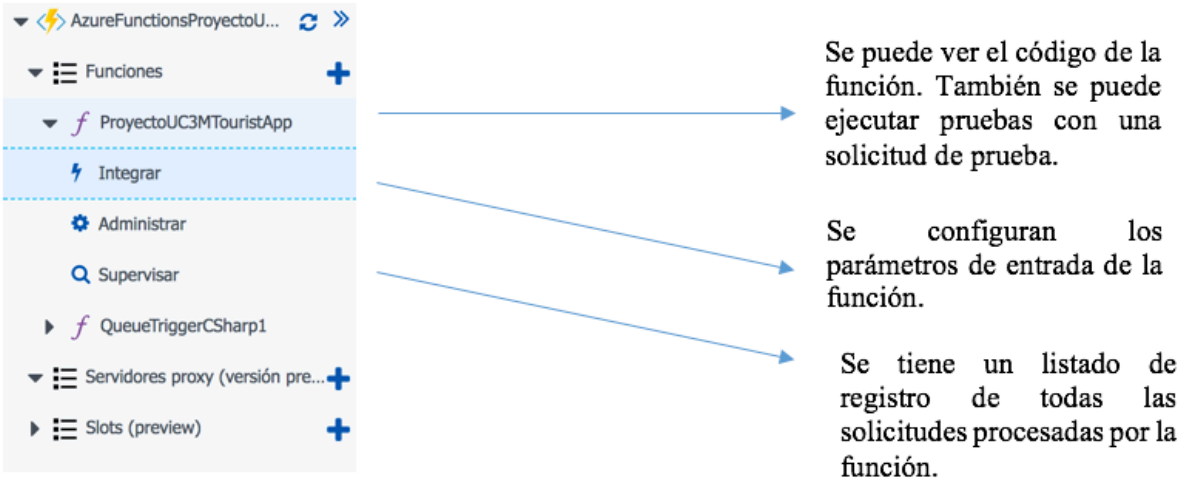


Figura 34: Menú Azure Function

Capítulo 4

EVALUACIÓN DE LA APLICACIÓN

En este capítulo se presentan las pruebas realizadas del sistema, que han sido realizadas desde las primeras etapas en el desarrollo hasta la versión final. A continuación, se describe el entorno de pruebas y después se detallan las mismas. Por último, se hará un pequeño análisis de los resultados obtenidos.

4.1 Descripción de las pruebas

Durante el desarrollo de la aplicación el código se ha depurado en un dispositivo Android que ha sido emulado desde el Android Studio.

- LG Nexus 5
  - SO: Android v5.1 (Lollipop)
  - CPU: Quad-Core Snapdrago 800.
  - GPU: Adreno 330.
  - GPS: Sí, con soporte A-GPS y GLONASS
  - Java: Sí, vía emulador Java MIDP
  - Dimensiones: 69,17 x 137,84 x 8,59 mm
  - Peso: 130g
  - Display: 4.7 pulgadas
  - Resolución: 1920 x 1080 píxeles (445ppi)

4.2 Pruebas realizadas

Se muestran las pruebas realizadas sobre la aplicación, en las cuales se han tratado de diseñar una aplicación robusta para evitar el número de casos de error.

ID: CP 1
Funcionalidad: Búsqueda de rutas. **Descripción:** Seleccionar opciones de preferencia a la hora de la búsqueda de una ruta. Verificar que la búsqueda funciona correctamente aun cuando no se introduce el campo nota o no disponemos en base de datos de una ruta con dicha puntuación. **Resultado:** Satisfactorio.

Tabla 2 : Caso de prueba 1– Búsqueda de rutas

ID: CP 2
Funcionalidad: Mostrado de resumen de ruta. **Descripción:** Pase de imágenes y título de cada uno de los puntos de la ruta en la vista de resumen. Además de la obtención de las opiniones de los usuarios de la base de datos. **Resultado:** Satisfactorio.

Tabla 3: Caso de prueba 2 – Mostrado de resumen de ruta

ID: CP 3
Funcionalidad: Mostrado de ruta en el mapa. **Descripción:** Verificar que el trazado de la ruta pasa por cada uno de los puntos del recorrido. **Resultado:** Satisfactorio.

Tabla 4: Caso de prueba 3 – Mostrado de ruta en el mapa

ID: CP 4
Funcionalidad: Funcionamiento del ActionBar. **Descripción:** Mostrado de los desplegables del listado de puntos de la ruta, de las opciones de búsqueda de lugares cercanos y de los puntos recorridos. **Resultado:** Satisfactorio.

Tabla 5: Caso de prueba 4 – Funcionamiento ActionBar

ID: CP 5
Funcionalidad: Reproducción de audios. **Descripción:** Mostrado de la barra inferior al seleccionar un punto para reproducir y parar el audio. Verificar que los botones de play y pause funcionan correctamente. **Resultado:** Satisfactorio.

Tabla 6 : Caso de prueba 5 – Reproducción de audios

ID: CP 6
Funcionalidad: Barra de progreso. **Descripción:** Verificar el funcionamiento de la barra de progreso que va en paralelo con la reproducción del audio. Además, parar esta tarea asíncrona al elegir otro punto de la ruta en el mapa. Lo mismo debe ocurrir cuando durante la reproducción de un audio se elige la opción de buscar un lugar cercano al dicho punto. **Resultado:** Satisfactorio.

Tabla 7 : Caso de prueba 6 – Barra de progreso

ID: CP 7
Funcionalidad: Búsqueda de lugares cercanos. **Descripción: Mostrado:** Verificar que en torno al punto seleccionado de la ruta se muestran los lugares cercanos del tipo seleccionado con su correspondiente logo descriptivo. A parte también verificar la opción de limpiar el mapa de los puntos cercanos encontrados. **Resultado:** Satisfactorio.

Tabla 8 : Caso de prueba 7 – Búsqueda de lugares cercanos

ID: CP 8
Funcionalidad: Cuadro de diálogo al seleccionar lugar cercano. **Descripción:** Verificar que al seleccionar un lugar cercano se nos muestra el cuadro de diálogo para poder buscar en internet información del punto seleccionado o trazar la ruta hacia ese punto. **Resultado:** Satisfactorio.

Tabla 9 : Caso de prueba 8 – Cuadro diálogo lugar cercano

ID: CP 9
Funcionalidad: Búsqueda de información en Internet. **Descripción:** Al pinchar en el cuadro de diálogo sobre buscar en internet, verificar que se abre el navegador con la búsqueda en Google de lugar seleccionado. **Resultado:** Satisfactorio.

Tabla 10 : Caso de prueba 9 – Búsqueda de información en Internet

ID: CP 10
Funcionalidad: Trazado de ruta hacia lugar cercano **Descripción**: Verificar que al seleccionar continuar sobre el cuadro de diálogo se traza la ruta sobre el mapa al punto cercano seleccionado. Además, comprobar que en la barra superior nos aparece la opción ruta para volver a cargar la ruta turística inicialmente seleccionada. **Resultado:** Satisfactorio.

Tabla 11 : Caso de prueba 10 – Trazado de ruta hacia lugar cercano

ID: CP 11
Funcionalidad: Guardado automático de los puntos recorridos. **Descripción:** Comprobar que tanto en base de datos y al seleccionar en el en la barra superior el resumen de los puntos, se muestran todos los que haya ido seleccionando el usuario. **Resultado:** Satisfactorio.

Tabla 12 : Caso de prueba 11 – Guardado de puntos recorridos

ID: CP12
Funcionalidad: Guardado de las respuestas de la API de análisis de texto. **Descripción:** Comprobar que guardamos el valor de la respuesta de dicha API, para mostrar en el listado de opiniones. **Resultado:** Satisfactorio.

Tabla 13 : Caso de prueba 12 – API análisis de sentimiento

Capítulo 5

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se hace un balance del trabajo realizado en el presente Trabajo Fin de Grado. Partiendo de la revisión de los resultados logrados, se comprueba que los objetivos fijados inicialmente han sido cumplidos y se exponen las conclusiones obtenidas. Para finalizar, se presentan los posibles trabajos futuros que se podrían desarrollar sobre la aplicación, de forma que aumentara su eficiencia y el número de funcionalidades.

5.1 Conclusiones

En este Trabajo de Fin de Grado se ha desarrollado una aplicación en Android cuyo principal objetivo es ofrecer un servicio de guía turístico.

Para la implementación de la aplicación ha sido necesario el estudio de diferentes campos dentro del desarrollo de aplicaciones en Android. En primer lugar, el análisis de las APIs de Google para trabajar con mapas, lugares y rutas. En segundo, la investigación sobre la implementación de bases de datos SQlite. En tercer lugar, la implementación de funciones asíncronas en el código para realizar ciertos desarrollos como la barra de progreso del audio. Y por último, el estudio de las diferentes APIs que ofrecen servicios de análisis de texto.

Originalmente la aplicación iba a mostrar rutas turísticas con varios puntos en un mapa, y dar la posibilidad al usuario de reproducir un audio en cada punto. Finalmente, tras conseguir ese primer hito se decidió añadir funcionalidades extra que aumentarán la potencialidad y el uso de la aplicación. Entre estas funcionalidades destacan la búsqueda de lugares cercanos entre los de una lista dada, así como el cálculo de ruta hacia ellos o la búsqueda automática en Internet de dicho lugar.

Con tiempo aún disponible para el desarrollo, se creó en la nube de Azure una plataforma que provee de un servicio de procesos de negocio de forma automatizada. Esta plataforma se basa en una Logic App (aplicación lógica) que se ejecuta cada un tiempo x configurado. Tras ejecutarse procesa los tweets para finalmente publicarlos junto con su firma en el canal de Slack correspondiente (malo, bueno o muy bueno) según el análisis de sentimientos del contenido del tweet.

También cabe añadir que cómo durante el desarrollo de cualquier proyecto, han surgido diversas problemáticas a la hora de implementar la funcionalidad o el flujo de ejecución que se tenía pensado. Sobre todo, problemas a la hora de sincronizar varios hilos de ejecución para que no se generase error, y conseguir así una aplicación robusta.

Y, por último, quisiera añadir que el desarrollo de este proyecto me ha servido para adquirir experiencia en el desarrollo de aplicaciones Android, así como manejar nuevos entornos de desarrollo cómo Android Sutdio o conocer plataformas de servicios en la nube y la potencialidad de Azure y Google ofrecen para la creación de aplicaciones.

5.2 Trabajo futuro

Debido a los límites de tiempo y al trabajo que supone, el presente proyecto tiene un amplio margen de mejora en diferentes aspectos, trabajando en ellos se podría ofrecer una versión más completa de la aplicación y con un mayor número de funcionalidades.

El primer objetivo sería ejecutar la aplicación en un dispositivo físico, ya que la actual aplicación únicamente se ejecuta en el emulador. Para conseguir esto sería necesario implementar una base de datos externa y conectarse a ella remotamente. Con este cambio la aplicación correría en un dispositivo, siendo el anterior el único problema, dado que durante el desarrollo se han ido probando de forma aislada cada uno de los módulos de los que consta la aplicación en un dispositivo físico.

Otras de las posible mejoras en las que se podría trabajar es en detectar la localización del usuario, así como ofrecer la aplicación en diversos lenguajes, siendo el del dispositivo el de por defecto. Detectar la localización no sería un desarrollo muy laborioso, ya que *Google Services* nos ofrece su librería para acceder a la localización del usuario. Y en cuanto al multilenguaje lo que habría que hacer sería acceder al idioma en el que está en el dispositivo y añadir todas las cadenas de texto en todos los idiomas al archivo strings. De tal forma qué, automáticamente en cuanto se detecte el idioma del dispositivo se deberían usar las cadenas de texto de ese idioma.

También algo que se podría implementar es un trazado de rutas dinámico. Es decir, se podría detectar si un número significativo de usuarios se desvía de la ruta ofrecida hacía puntos cercanos a esta. Si esto ocurre se podría modificar la ruta por defecto para que pase por los puntos hacía los que los usuarios se han desviado.

Y, por último, se podrían asociar fotos de los usuarios a cada uno de los puntos de la ruta, para que otros puedan ver las fotos que anteriores turistas realizaron en dicho punto. De esta manera tendríamos una galería de fotos para cada uno de los puntos, pudiendo ser estas comentadas o puntuadas.

Capítulo 6

PRESUPUESTO E IMPACTO SOCIO-ECONÓMICO

Este capítulo incluye un presupuesto detallado con el coste estimado de lo que podría haber desarrollado este proyecto. A parte se hace una análisis detallado el impacto socio-económico que produciría lanzar dicha aplicación al mercado.

6.1 Presupuesto

Seguidamente se muestra el coste total para realizar la implementación de la aplicación.

Recursos HW:

Recurso	Importe
Ordenador portátil:	1400 €
Subtotal HW:	**1400 €**

Recursos SW:

Recurso	Importe
Paquete Office 365 Universidad	0 €
Android Studio	0 €
API Meaning Cloud	0 €
Azure	130 €
APIs de Google	0 €
Subtotal SW:	**130 €**

Recursos Humanos:

Recurso	Importe
Hora laboral Programador	45 €/h

Tareas:

Fase 1: Planificación.

Tarea	Duración/persona (Días de 3 h)	Importe
Estudio de las APIs a usar	15 Días	675 €
Planificación y análisis de los requisitos	3 Días	135 €
Estudio de las tecnologías a utilizar	20 Días	900 €
Subtotal:		**1710 €**

Fase 2: Desarrollo.

Tarea	Duración/persona (Días de 3 h)	Importe
Pruebas iniciales y configuración de Android Studio	5 Días	225 €
Implementación de la aplicación	40 Días	2025 €
Pruebas unitarias	8 Días	360 €
Implementación de la plataforma en Azure	7 Días	315 €
Pruebas unitarias	3 Días	135 €
Subtotal:		**3060 €**

Fase 2: Documentación.

Tarea	Duración/persona (Días de 3 h)	Importe
Memoria	35 Días	1575 €
Presentación	15 Días	675 €
Subtotal:		**2250 €**

Coste total:

Subtotal HW	Importe
Subtotal SW	130 €
Subtotal Planificación	1710 €
Subtotal Desarrollo	3060 €
Subtotal Documentación	2250 €
Subtotal:	**7150 €**

6.2 Impacto socio-económico

En la actualizad el mercado ya cuenta con aplicaciones similares a la desarrollada en este proyecto, por lo que poder hacerse un hueco entre estas no es tarea fácil. Para poder conseguirlo es necesario ofertar funcionalidades nuevas o de diferente forma.

A continuación se detallará el impacto tanto social como económico deseable a conseguir por el lanzamiento de la aplicación. No de la desarrollada en este proyecto, pero si de una que cuente con sus funcionalidades, además de otras que no se han podido implementar por tiempo y complejidad.

Impacto social

En la actualidad el uso del móvil es cada vez más indispensable, sobre todo cuando viajamos a ciudades o países que no conocemos. El lanzamiento de esta aplicación va destinada al ámbito turístico, con la idea de que los viajeros puedan conocer de forma fácil las ciudades realizando rutas con temáticas diferentes (culturales, gastronómicas…). De forma que sin la ayuda de nadie y con total desconocimiento, el viajero pueda disfrutar de todos los rincones míticos de la ciudad.

Las rutas turísticas solo es la punta del iceberg de la aplicación, además de eso, los usuarios agradecerían poder consultar de forma fácil otro tipo de servicios, como localizar rápidamente las estaciones de alquiler de bicis, o de coches sin salir de la aplicación, con esto, los usuarios de la aplicación podrán desplazarse con mucha más facilidad por la cuidad. A parte de estos servicios, también se ha pensado en mostrar de forma fácil los diferentes eventos que se celebren en la ciudad durante las fechas en las que los viajeros deseen buscar. Sin olvidarse de los propios usuarios, ya que estos podrán alimentar la aplicación con sus opiniones y sugerencias de los servicios que esta ofrezca. Hasta la posibilidad de introducir rutas personalizadas con nuevos puntos turísticos que en un principio no se tuvieran en cuenta.

Con este conjunto de funcionalidades se pretende potenciar el turismo en las ciudades con la comodidad de hacerlo sin salir de una aplicación, o la intermediación de terceros, ya sean oficinas de turismo o engorrosos folletos de publicidad, únicamente

con la aplicación y los propios usuarios. Esta forma de turismo podría suponer una pequeña revolución, principalmente para las personas con menor conocimiento de idiomas, o poca iniciativa para la aventura por temor al desconocimiento del lugar o por el agobio de tener que planear y buscar los lugares que visitar.

Impacto económico

A día de hoy el sector turístico tiene un impacto considerable en las economías de casi cualquier país. Si tomamos como referencia Madrid, este sector supone para la capital española una inyección de más de 6000 millones de euros al año (datos 2016) [EUROPAPRESS].

A las personas cada vez les gusta más viajar, y el lanzamiento de una aplicación que facilite la experiencia de realizar turismo, unificando todo un conjunto de funcionalidades que interesen al turista, sin duda haría aumentar el número de viajeros por diversos motivos, ya sea por no tener que preocuparse de planear viajes o no perder el tiempo en busca de terceros que te aconsejen que visitar.

El lanzamiento de una aplicación de este tipo captaría la atención de los diferentes sectores que competen el sector turístico, como pueden ser el hotelero, transportes o restauración. Dando la opción que estos puedan publicitarse o aparecer en la aplicación para dar a conocer al turista su ubicación, ofertas u opiniones.

Se puede concluir que, el impacto económico de lanzar una aplicación que posibilite al usuario de hacer uso de servicios de diferentes sectores, aparte de usarla como una aplicación de ocio, supondría un impacto que haría aumentar a día de hoy los ingresos procedentes del turismo, casi con total certeza.

Capítulo 7

MARCO REGULADOR

Este apartado tiene por finalidad establecer el marco regulatorio referente a los estándares técnicos aplicados en el proyecto presente, sobre tecnología desarrollada, implantada, lenguajes de programación o herramientas utilizadas.

Tipo y funcionalidad del software implementado

Aplicación móvil para sistema operativo Android, cuya funcionalidad principal se basa en un asistente turístico. La aplicación permite buscar rutas turísticas formadas por varios puntos y da la posibilidad al usuario de leer y escuchar la información de cada uno de los puntos de la ruta. A parte ofrece funcionalidades extra, como de la búsqueda de lugares de cercanos y recoger la opinión del usuario mediante voz.

Consideraciones generales de desarrollo del software que se han tenido en cuenta

- El software implementado cuenta con tiempos de respuestas razonables en los procesos tanto en los que se hace uso de servicios externos como los propios que han sido desarrollados para ofrecer una funcionalidad determinada.
- El software posee programación ordenada y documentada.
- El software posee interfaces amigables, fáciles de usar y entendibles por el usuario final.

- Al tratarse de una aplicación demo se han seleccionado los planes gratuitos para el uso de los servicios externos.

- Implementación de bases de datos local para probar y ejecutar la aplicación.

Lenguajes de programación utilizados

Para el desarrollo de este software ha sido necesario el uso de los siguientes lenguajes de programación.

- Java para la implementación de toda la lógica de la aplicación.
- *XML* para el diseño de cada una de las actividades de la app.
- C# para la implementación de las *Azure Functions*.

Servicios externos utilizados

Para poder añadir las funcionalidades que trabajan con rutas y mapas ha sido necesario hacer uso dos *APIs* de *Google*.

- *API Direcctions* para representar rutas en el mapa.

- *API Google Places* para buscar lugares cercanos.

Para el análisis de sentimientos de las opiniones de los usuarios introducidas desde la aplicación se ha usado:

- *MeaningCloud*.

Para el análisis de sentimientos de los tweet de los usuarios se ha el servicio de *Azure* de análisis de sentimientos.

Plataformas para el desarrollo utilizadas

- Para el desarrollo de la aplicación se ha utilizado Android Studio en su versión 2.2.2.

- Para la implementación de la *Logic App* se utilizado la plataforma en la nube de *Microsfot*, *Azure*.

- Para el manejo de base de datos se ha utilizado una extensión del navegador Firefox *SQLite Manager* versión 0.8.3.1

APPENDIX IN ENGLISH

THANKS

To my classmates and friends of the university with I have spend moments into and out of the university, specially to my best friend and classmate Gabriel with whom I have traveled this long road that is already at an end, to the hope that new doors will open to achieve and surpass new goals and challenges.

To my parents, Ana and Cristóbal, because without them would not have been possible to get here, they have always supported me to give my best in the most difficult moments throughout these years. Without forgetting my girlfriend, Beatriz, who has been a importante pillar in my last course.

By the last, to my tutor David, who have helped me during the project development, and my work companions, who help me to nd help me grow both professionally and personally.

INTRODUCTION

Nowadays, phones are essential to our actual life, especially when we travel to countries or cities that we do not know very well and we have to know how to arrive to a determinated place. That need gave me the idea to create an application which will provide us a complete service to know how to visit the famous places in a city.

After sharing the idea with my Tutor and take his approval, I begun this proyect. First of all, I had to check how was the application´s market with a similar service and I took an idea to incorporate the functionality of tours. The rest of the services added to the aplication were made to improve and complete the application that already existed.

Secondly, I had to define the functionalities like a tour guide able to show routes in the map and give the information of every point of selected tour though audio note. Moreover, after completing the tour, the user has the possibility of introducing an opinión about the service. As to design requirements, the idea was to create an application with a simple graphic interface, intuitive and nice too.

Because of my inexperience in Android´s applications, the begginnings were not easy. The first problema that I had was to find an Android operating system version to compile a project with Google maps in Android Studio. After some tests, the project was created with 5.0 version and the emulator to implement it with 5.1.

Then I started with the Google APIs study and tests to learn as these services work, specially how to deserialize the answers that they return. After, I started the implementation of the application, specific of the main funcionality, the displayed routes, which included the call to Google Directions API. When I got this milestone, I did the first data base, routes.sqlite, this table have information about the route (name, points number...). The other tables are PointTracked.sqlite, that save user information and the last NamePointext.sqlite, this save the description each points of route.

To complete the main activity was only the reproduction and control of audio files, this complemented almost all the basic functionality. Only it was to implement the use of the analysis sentiment, for it was necessary to save each point travelled for the user. Whe the user will complete all points he could to introduction his opinion for the analysis with Meaning Cloud API.

Once all the requirements have been fulfilled, one new funcionality was added, the search of nearby places of interest and the calculation of route to them, for this we use Google Places. On the other hand, I added new activities which improved user experience, as well as the possibility of searching for routes based on preferences set by the user and an activity to display a set of information of the pre-selected route.

When the aplication was stable, I implemented in Microsoft Azure an automated business process platform. This is able to process the tweets and publish on a Slack channel according to the content of these bad, good or very good.

Finally add that the objectives marked at the beginning have been fulfilled, gaining experience and management in the development of Android applications. In addition to knowing Cloud platforms like Azure, which opens a range of possibilities which increases the number of functionalities that the application can offer.

CONCLUSIONS

This chapter makes a balance of done work in the present final degree project. Based on the review of the results achieved, it is verified that the goals initially set have been met and the conclusions drawn are presented. Finally, the future works to make better the application and increase the funcionalities number also these are comented.

In this project has developed an application in Android whose main objective is to offer a tour guide service. The study of differents fields of the Android application development has been necesary to do this application. First, the Goolge APIs analysis to work with maps, routes and places. Second, the implementation async task in the code to make certain developments like the audio progress bar. Lastly the APIs study to analyze feelings.

At first, the application was going to show the tourist routes with several points in a map, and give to the user the posibility to play an audio in each point. When I got this objetive I decided to add more extra funcionalities. Between these were the search nearby places and the calculate of the route to these points.

With time to spend for the development, I built in the Azure cloud a platform that provides business process automated. This platform is done with a Logic App, this runs every a configurated time. The action next processes the tweet fo finally these were posted in the corresponding Slack channel (bad, boog or very god) depending of the sentiment analysis of thetweet content.During the project development several problems have arisen when implementing the functionality or the execution flow that was thought, specially sync problems with threads.

Finally, I want to say that this project has served me to get experience in the application development, and to know development environments as Android Studio, or new platform as Azure or Google Cloud.

FUTURE WORK

The present project has a wide margin of improvement in different aspects, working in them could offer more complete version of the application and with more functionalities.

The main objective was to run application in physical device, to achieve this would be necessary to implement an external database and connect to it remotely. Other than the improvements would be to detect the user location or implement multilanguage. To detect the location google offers Google Services. On the other hand, I would have to do the language in which it is on the device and add all strings of text in all languages to the strings file.

Also, the application could have a dynamic traced route, that is, it could be detected if a significant number of users deviate from the route offered to points near this one. If this happens you could modify the default route

Finally, it would be interesting that the users could associate photos for each points of the route, and create a photo gallery so that other users can rate or comment the photos of previous users.

SOCIO-ECONOMIC IMPACT

Nowadays, there are similar applications to the ones I have developed in this proyect so it is difficult to introduces these in the business. We have to offer aditional funcionalities or in different ways.

Then, I am going to show in a deeper way the impact econominic and social that I hope to get with this application.

Social impact

Currently, people use much more the móvil pone, especially when we travel to different countries. This turist applicacation is thinking with the idea to offer more information and routes (culturals, gastronomic…) without a touris guide. So, people can see the city byself.

Users would appreciate being able to get other kinds of services, such as quickly locating bicycle or car rental stations without leaving the application. It offers an easier and quickly way to move around the place.

Besides, it can show easily the different events that take place in the city during the desired dates. Apart of this, users have the possibility to show their opinions and suggestions.

It wants to potentiate tourism easily, without brochures or tourism offices. This form of tourism could be a small revolution, mainly for people with less languages knowledge or little initiative to travel, because of they are afraid of an unknow city.

Economic impact

Tourism sector has a considerable impact on the economies of almost countries. For example, Madrid gets more than 6000 million of euros (2016 data).

An application that facilitates the experience of tourism, unifying a whole set of features that interest the tourist, would certainly increase the number of travelers for various reasons. Firstly, you do not have to worry about planning trips or do not waste your time looking for a person who advises you to visit.

An application of this type would capture the attention of the different sectors that compete in the tourism sector, such as hoteliers, transports or restaurants. they can advertise or appear in the application to let the tourist know their location, offers or opinions.

For all this, this application could contribute significantly to increase economic benefits from tourism.

GLOSARIO

Termino	Descripción
Spinner	Menú desplegable que muestra una lista de opciones.
ListView	Lista de opciones para la elección del usuario.
SQLite	Tipo de base de datos para Android.
Cursor	Esta interfaz proporciona acceso de lectura y escritura aleatorio al conjunto de resultados devuelto por una consulta de base de datos.
Ogg	Formato reducido para archivos de audio
XML	Extensible Markup Language
MediaPlayer	Objeto para trabajar con archivos de audio en Android.
HashMap	Colección de objetos de mapa, cada uno identificado por una clave.
SqliteManager	Extensión de Firefox para trabajar con base de datos SQLite.
AsyncTasck	Interfaz para la implementación de flujos de código asíncrono.
ADVManager	Proporciona una interfaz gráfica de usuario para crear y administrar dispositivos virtuales.
Azure	Plataforma en la nube de Microsoft.
VR	Realidad virtual.
AR	Realidad aumentada.
Machine-Learning	Aprendizaje automático
API	Interfaz de programación de aplicaciones.
IA	Inteligencia Artificial.

REFERENCIAS

[CISCO] (2017*).* El tráfico global de datos móviles se multiplicará por siete entre 2016 y 2021. En *Globalnewsroom Cisco*. Recuperado de http://globalnewsroom.cisco.com/es/es/release/El-tráfico-global-de-datos-móviles-se-multiplicará-por-siete-entre-2016-y-2021-2475531

[VPNMENTOR] Tendencias de internet, estadísticas y datos en los Estados Unidos y el mundo para 2017. En *Vpnmentor*. Recuperado de https://es.vpnmentor.com/blog/tendencias-de-internet-estadisticas-y-datos-en-los-estados-unidos-y-el-mundo/

[SOURCEANDROID] Recuperado de http://source.android.com/

[DEVELOPERANDROID] Recuperado de http://developer.android.com/about/dashboards/index.html

[GOOGLEDEVELOPERS] Recuperado de https://developers.google.com/

[MEANINGCLOUD] Recuperado de https://www.meaningcloud.com/developer/sentiment-analysis/doc/2.1/response

[MICROSOFT] *Docs Microsoft*. Recuperado en https://docs.microsoft.com/

[STACKOVERFLOW] Recuperado de http://stackoverflow.com/

[GOOGLEAPIDIRECTION] Recuperado de https://developers.google.com/maps/documentation/directions/?hl=es

[GOOGLEPLACESAPI] Recuperado de https://developers.google.com/places/?hl=es

[PWERDATA] Recuperado de http://www.powerdata.es/big-data

Arrabales, Raúl (2016). Deep Learning: qué es y por qué va a ser una tecnología clave en el futuro de la inteligencia artificial. En *Xataka*. Recuperado de https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial

Bejerano, Pablo G. (2017). Diferencias entre machine learning y Deep learning. En *Blogthinkbig*. Recuperado de https://blogthinkbig.com/diferencias-entre-machine-learning-y-deep-learning/

[MICROSOFTAZURE] (2107) Qué son las Logic App. En *Docs Microsoft*. Recuperado de https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-what-are-logic-apps

[MICROSOFTAZURE] (2017) Ejemplos y escenarios comunes de Azure Logic Apps. En *Docs Microsfot*. Recuperado de https://docs.microsoft.com/es-es/azure/logic-apps/logic-apps-examples-and-scenarios

[STATISTA] (2017). Global mobile OS market share in sales to en users from 1st qurter 2009 to 1st quarter 2017. En *Statista*. Recuperado de https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[INVERTIA] (2017). El negocio del big data alcanzará los 5.500 millones en Europa en 2018. En *Invertia*. Recuperado de https://www.invertia.com/es/-/el-negocio-del-big-data-alcanzara-los-5-500-millones-en-europa-en-2018

Poyatos, Juan M. Big Data y el sector de la salud: el futuro de la sanidad. En *Poyatos Díaz*. Recuperado en http://poyatosdiaz.com/index.php/big-data-y-el-sector-de-la-salud-el-futuro-de-la-sanidad

[EUROPAPRESS] (2016). El sector del turismo tiene un impacto económico en Madrid de 6.000 millones. En *Europapress*. Recuperado de http://www.europapress.es/turismo/destino-espana/turismo-urbano/noticia-sector-turismo-tiene-impacto-economico-madrid-6000-millones-20160427170104.html